On the solution of non linear systems.

Daoud, D. S.

Award date:
1981

Awarding institution:
University of Bath

ON THE SOLUTION OF NONLINEAR SYSTEM

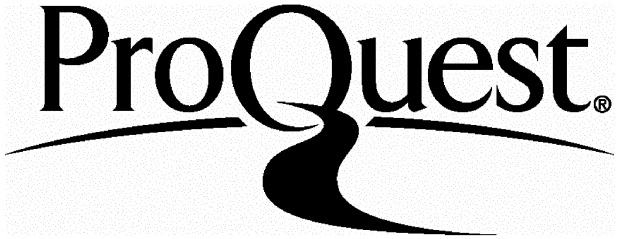
submitted by D.S. Daoud, M.Sc., G.I.M.A.

for the degree of Ph.D.

of the University of Bath

1981

D.S. Daoud



To My Parents
With Love and Gratitude

To My Sincere Friend
My Big Brother "Monkith"

And to All of My Sisters
and Brothers

Acknowledgements

It is pleasure to acknowledge the help and encouragement of my supervisor Mr. F.B. Ellerby.

I am indebted to my supervisor for his help during the early stages of familiarising myself with earlier work in this field and his friendly advice.

Also it is pleasure to acknowledge the help and the advice from the programmers and the operators in the computer centre.

Abstract

We consider here two classes of non-linear systems giving different degrees of non-linearity. In both cases the systems arise from finite difference discretisations of non-linear elliptic partial differential equations.

Our solution methods can also fit into two categories - linearisation and non-linearisation techniques - and in our studies we have pursued three main objectives.

1) For mildly non-linear systems we generalise certain iterative techniques from the solution of linear systems to the solution of non-linear systems with symmetric Jacobians. We are especially concerned with the effect of preconditioning of the equations here.

2) We consider the use of bidiagonalisation on non-linear systems, using preconditioning in two ways and considering both classes of non-linear problem.

3) We solve the laminar flow problem and assess the effects of multigrid acceleration on non-linear S.I.P. techniques.

List of Contents

CHAPTER 0 Introduction

0.1 Preface

In this thesis we consider the numerical solution of nonlinear systems of equations. $F(x) = 0$, where $F:D \subset R^n \longrightarrow R^n$, and $F'(x)$ is a large sparse symmetric (or non symmetric) matrix, generated from the application of finite difference approximations.

Solving non linear system is unlike solving linear systems, where direct methods for some problems is feasible, so we must focus on iterative techniques which can be categorised as follows:

1. Linearization methods
2. Non linearization methods

The use of the linearization method, which includes all the applicable techniques to linear systems (e.g. conjugate gradient method, S.O.R. method and A.D.I. method) in solving a system of non linear equations requires the evaluation of the non linear operator either at a prior fixed number of iterations or until convergence of the prelinearized system is achieved (e.g. A.D.I. method, see (Y2)).

But some of the nonlinearization methods require the evaluation of the Jacobian of the non linear system (e.g. non linear conjugate gradient method (D2), preconditioned non linear conjugate gradient method (C10)).

We investigate the numerical solution of two categories of non linear operator.

C_1 - $Ax = F(x)$ (A is a matrix of constant coefficient and F is non linear operator)

C_2 - $A_{(x)}x = b$ ($A_{(x)}$ is a matrix, each component is a non linear operator of the solution vector x)

An example of C_1 is the system generated from the approximation of the mildly non linear equations (H8), and an example of C_2 is a laminar flow problem (Y2).

0.2 Review of related literature

An early work on mildly non linear elliptic problems is due to Bers, 1953 (B4), who considered a finite difference equation

$$\Delta u = F(x,y,u,u_x,u_y)$$

Bers proved the convergence and uniqueness of the corresponding difference equation.

Douglas in 1961 (D10) introduced a modification of the usual A.D.I. method to solve Dirichlet problems for mildly non linear equations.

In 1965 Parter (P6) proved the convergence of the solution of the finite difference equations arising from an approximation of the mildly non linear equations, Greenspan and Parter (G7) continued the work considering another class of iterative technique to solve these problems.

Ortega and Rockoff (O3) generalized some linear iterative techniques which had been considered by Varga (V1) and applied them to solve mildly non linear elliptic equations.

In 1975, Hageman and Porsching (H8) solved a mildly non linear elliptic equation using non linear point (block) Gauss Seidel and non linear point (block) S.O.R. methods.

Chavette and Stenger (C3) solved a spherically symmetric case of $\Delta u = u - u^3$, Jain and Kadalbajo (J2) used a method of dynamic programming to solve a mildly non linear elliptic problem defined over an irregular region.

In 1979, Noor and Whiteman (N12) used finite element approximation to solve mildly non linear elliptic equations and proved the equivalence in a Hilbert space of variational and weak formulations of these problems. Meyer-Spasche (M11) proved some theorems concerning the convergence of the approximation of our problems with Dirichlet boundary conditions. Newton's method is the most popular technique and it has been widely used to solve non linear elliptic equations, it can be combined with any linear iterative technique (as a secondary iteration).

Ortega and Rheinboldt, in 1967 (O4), extended the results of Greenspan and Parter (G7), and incorporated them into a general theory for a broad class of monotone iterative techniques such as Newton Gauss Seidel technique. In 1971 McAllister (M5) applied Newton's method to solve the system generated from the application of the finite difference approximation on uniformly elliptic operator defined over a certain region R.

In 1971, Schryer (S3), applied Newton's method to solve $Lu = f(x,u)$, where L is a linear elliptic operator and f is a convex monotone increasing function, and in 1972 Schryer (S4) solved the same system, when f is a monotone increasing function.

Madson in 1973 (M1) solved non linear equations using Newton's method adopting the evaluation of the Jacobian in the iteration method. Dennis and More in 1977 (D7) proposed a successful modification of a quasi Newton method to a system of non linear equations. In 1978 Sherman (S8) studied the local rates of convergence of Newton's method in the solution of a system of non linear equations.

The variety of the iterative techniques is a consequence of the variety of non linear operators, and the problems of error, stability, convergence and "starting" have been subjects of research. Bramble and Hubbard in

1963 (B12) presented a theorem on error estimation for a certain class of finite difference approximation applied to approximate second order uniformly elliptic operator. Freudenstein and Roth (F5) studied the kind of starting iterative vector to solve non linear equations.

In 1964, Gunn (G8) solved $\nabla.a\nabla u = f$, using a semi explicit A.D.I. technique, Young and Wheeler (Y2) applied the Peaceman and Rachford A.D.I. method to the solution of the laminar flow problem. In 1966 McAllister (M4) studied the application of the difference method to solve the quasilinear uniformly elliptic partial differential equations, with Dirichlet boundary conditions. Concus in 1967 (C6) solved the minimal surface equation using point S.O.R. method and in 1969 (C7) he solved the same problem using block S.O.R. method, proving that block S.O.R. method converges more rapidly than point S.O.R. method.

Porsching in 1969 (P9) modified non linear Gauss Seidel and Jacobi methods to solve the Network problem.

In 1971, Brown and Gearhart (B23) studied a certain class of method to find an additional solution of non linear system, Voigt (V3) considered the order of convergence of an iterative procedure to find a zero of non linear function.

Stepleman (S14) considered a class of difference approximations for a second order quasilinear elliptic operator with mixed derivative and proved the existence of the solution when the right hand side vector is bounded.

In 1974 Karachersk and Lyashko (K1) investigated and constructed difference schemes solving 1st, 2nd and 3rd boundary value problems for quasilinear elliptic equations when the coefficients are discontinuous of the 1st

kind with respect to the independent variables x on hyperplanes parallel to the coordinates hyperplanes. Keller in 1975 (K2) studied the non linear problems, in the abstract form $F(x) = 0$, and families of approximating problems in the form $F_h(x_h)$, also he studied, briefly, the relation between "Isolation" and "Stability".

Dey, in 1976 (D8), proposed the use of a perturbed non linear Jacobi iteration to solve a system of non linear equations.

Moore in 1977 (M13) studied the global convergence of iterative schemes of the form $x_{k+1} = x_k - (P_k(x_k))^{-1}F(x_k)$, with special emphasis on the Newton's Gauss Seidel method, where F is continuously differentiable and convex on all of R^n and P_k^{-1} is a non negative subinverse of $F'(x)$ for each x in R^n. He also generalized some of Varga's results (V1) and improved some of Greenspan and Parter's results (G7).

Mittleman in 1977 (M12), solved the Dirichlet problem for a class of quasilinear elliptic equations approximated by the simplest finite element method.

Thews, in 1979 (T2) studied the existence and multiplicity question for non linear elliptic boundary value problems of the type

$$Lu(x) = P(x,y) \quad \text{for } x \in G$$

and $u(x) = 0 \quad \text{for } x \in \partial G$

where L is a uniformly elliptic formally self adjoint operator of second order and G is sufficiently smooth bounded domain in R^n. In 1980 Lipitakis and Evans (L7) presented a new extendable LU sparse factorization solving non linear elliptic difference equations with applications on non linear elliptic boundary value problems of 2 and 3 dimensions.

0.3 Introduction to the present work

This thesis can be divided into two parts, corresponding to the case in which the Jacobian of the non linear system is:

1. Symmetric positive definite
2. Non symmetric

In chapter 1 we describe some preconditioning techniques. Some of these techniques are derived from other iterative methods like SSOR method (see Y1) and A.D.I. method (see V1), others from the approximate factorization schemes due to Meijerink and Van der Vorst (M17), Dupont et al (D14).

We consider in chapter 1 the approximate factorization by H.L. Stone (S15) and its modification to produce a symmetric approximate matrix factorization by A. Bracha-Barak and P. Saylor (B10).

In chapter 2, we deal with the finite difference approximation for the mildly non linear elliptic equation. The approximated system is of the form $Ax = F(x)$ (of symmetric and positive definite Jacobian). We propose two iterative techniques to solve the system $Ax = F(x)$, the linearized S.I.P. and the non linearized S.I.P. techniques. The linearized S.I.P. technique is the simplest application of S.I.P. which requires the evaluation of the non linear operator in order to have a linear system at a prior chosen fixed number of iterations.

The non linearized S.I.P. is a new iterative technique which we propose. This method adapts the evaluation (or any approximation) of the Jacobian (compared, say, with Newton's method) and provides a fixed value throughout the iteration.

In chapter 3 we describe a class of variational iterative methods for solving symmetric linear systems, here we generalize some of the important properties (some of those properties have been proved by R. Chandra (C4) for linear systems) of the system of non linear equations using non linear conjugate gradient method (D2) and also we obtain general error bounds for non linear systems. Also we motivate and study the error bounds of the preconditioned non linear conjugate gradient method (using the approximate matrix factorization (B10)).

In chapter 4 we present the numerical results from solving three model problems (mildly non linear elliptic equations of symmetric and positive definite Jacobian solved by Hageman and Porsching using the non linear point (block) Gauss Seidel method, the non linear point (block) S.O.R. method), also nonlinearized S.I.P., linearized S.I.P., linear (nonlinear) conjugate gradient and preconditioned non linear conjugate gradient methods.

We proposed total number of arithmetics, size of storage units, total number of iterations and time in order to measure computational complexity. The numerical results show that nonlinearized S.I.P. is an optimal method, among all of the proposed iterative methods, with regard to the suggested comparison factors.

We compare the results of using nonlinearized S.I.P. with the results of using the methods of (H8), it shows that nonlinearized S.I.P. is a more economical method (with regard to the number of iterations and time).

In chapter 5 we consider the iterative technique "Bidiagonalization method" (see (P1)) to solve a non linear system, with non symmetric Jacobian, by linearization.

We solved two model problems, approximated by finite difference approximations, problem 1 (see C10) generates a non linear system of the form $Ax = F(x)$ where A is a matrix of constant coefficient and F is the non linear operator and problem 2 (see Y2) generates a non linear system of the form $A(x)x = C$, where $A(x)$ is a system each component is a non linear function of x and C is a constant vector.

The numerical results, from using linearized bidiagonalization show the inefficiency of this method (with regard to both the time and the number of iterations). We proposed two preconditioning bidiagonalization techniques to accelerate the convergence (and consequently reduce the total number of iterations and time), using the Stone approximate matrix factorization for non symmetric matrix (S15).

These preconditionings are:

1. Post preconditioned bidiagonalization technique
2. Pre preconditioned bidiagonalization technique

(The preprecconditioned bidiagonalization technique is equivalent to the technique proposed by Meijerink and Van der Vorst (M8)).
Using post preconditioned and pre preconditioned bidiagonalization techniques to solve problem 1 and problem 2, the numerical results show the efficiency of post preconditioned bidiagonalization method, with respect to the time and total number of iterations compared with the pre preconditioned bidiagonalization and with the bidiagonalization method.

Also we compare the post preconditioned bidiagonalization technique with the non linearized S.I.P. method (using Stone approximate factorization (S15)), the numerical results show the efficiency of the nonlinearized S.I.P. technique.

In chapter 6 we give a comparison of solving the laminar flow problem, problem 2 (see (Y2)), using a multigrid method with non linearized S.I.P. as a relaxation technique, we compare with the results of chapter 5, the numerical results to show that the multigrid method produces 50-60% reduction in the total number of iterations and time.

Finally, in chapter 7, we summarize the numerical developments in this thesis and give suggestions for future work.

CHAPTER 1 Matrix Factorization

1.1 Introduction

Consider the system of linear equations

$$Ax = b \qquad 1.1.1$$

where A is an N x N symmetric positive definite matrix. We can scale the linear system 1.1.1 using non-singular matrix Q and we have the following equivalent system

$$A'x' = b' \qquad 1.1.2$$

where

$$A' = Q^{-1}AQ^{-T}, \quad b' = Q^{-1}b \quad \text{and} \quad x' = Q^{T}x$$

Clearly A' is symmetric and positive definite. One can prove (see A4), when solving 1.1.1 by classical iterative methods, that the bound on the rate of convergence is a monotone decreasing function of k(A). Therefore one way to improve the rate of convergence is to choose Q to decrease the condition number of the iteration matrix. More precisely, if Q is chosen such that

$$k(A') < k(A)$$

then the algorithm may converge faster asymptotically of the preconditioned problem 1.1.2 than for 1.1.1. Since Q is non-singular, $M = QQ^{T}$ is positive definite. Conversely, any symmetric positive definite matrix M can be written as a product QQ^{T}, where Q is non-singular (see Y1). Thus the question of choosing an appropriate non-singular matrix Q is equivalent to the question of choosing a splitting of the matrix A of the form

$$A = M - R$$

Most iterative methods are based on that idea.

Associated with this splitting is an iterative method

$$Mx_{n+1} = (M - A)x_n + b = Rx_n + b \qquad 1.1.3$$

or $x_{n+1} = x_n + M^{-1}(b - Ax_n)$

The more M^{-1} ressembles A^{-1} the faster the method will converge. On the other hand we have to solve the equation

$$M\Delta x_n = b - Ax_n$$

during every iteration, so there is a requirement for M to be, in some cases, "triangle".

In 1967, D.J. Evans (E3) proved that the asymptotic convergence of many standard iterative methods for the solution of linear systems depend inversely on the P-condition number of the coefficient matrix, in addition to that he showed that calculating a pre-conditioning factor, to minimize the P-condition number, is computationally feasible and he discussed the application of this idea to the method of simultaneous displacements iterative technique.

O. Axelsson, 1974 (A6), proved that for semi iterative method the number of necessary iterations required to achieve a certain accuracy is directly proportional to the square root of the spectral condition number k(A).

1.2 Iterative Techniques and Matrix Factorization

Consider the system of linear equations

$$Ax = b \qquad 1.2.1$$

Such a system results from the application of finite difference approximations to linear self adjoint elliptic boundary value problems

$$L(u) = \sum_{i=1}^{p} \frac{\partial}{\partial x_i} (a_i(x) \frac{\partial}{\partial x_i} u) + c(x)u = f(x), \quad x \varepsilon D$$

and $u(x) = g(x), \quad x \varepsilon \partial D$ 1.2.2

(and also with Neuman and mixed boundary conditions),

where $a_i(x)$ are strictly positive functions and $c(x) \leq 0$.

Solving a system of linear equations by an iterative method was extensively well known through the past sixty years. For instance an important technique for solving a system of linear equations is the S.O.R. method (an example of 'fixed point' iteration).

$$x_{i+1} = x_i - \omega(D - \omega L)^{-1}(Ax_i - b) \qquad 1.2.3$$

where $A = D - L - U$, D is a diagonal matrix, L and U are strictly lower and upper triangular matrix respectively (see D. Young (Y1), R. Varga (V1), Ortega and Rheinboldt (O4)). The matrix $(D - \omega L)^{-1}$ which appears in 1.2.3 in fact reduces k(A) (see (A6), (O4), (V1) and (Y1)).

Another iterative method used to solve a system of linear equations is a factorized two step method such as the A.D.I. method. The matrix A can be split as (see (A6), (O4), (V1), (W1) and (Y1))

$$A = B + H_x + H_y \qquad 1.2.4$$

Then an A.D.I. iterative method to solve 1.2.2 is

$$\begin{aligned} (B + H_x)x^{\ell+\frac{1}{2}} &= -H_y x^{\ell} + b \\ (B + H_y)x^{\ell+1} &= -H_x x^{\ell+\frac{1}{2}} + b \end{aligned} \qquad \ell = 0,1,2,\ldots \qquad 1.2.5$$

Unless the set of equations is small, S.O.R. converges rather slowly, but because its properties are well understood it is reliable and popular, but the difficulty with A.D.I. is that although the method is

very fast for a certain class of problems this class is rather small.

For more about the difference between A.D.I. and S.O.R. methods see (W1).

Because of the inadequacy and difficulties of the A.D.I. and S.O.R. methods, other methods to solve 1.2.1 have been searched out; one of such methods is based on approximate factorization, (i.e.) find L and U, such that LU satisfies LU = A + R. We can view this approach of course as one in which

$$A = LU - R \qquad 1.2.6$$

where LU is symmetric positive definite and "close" to A in some sense, i.e. R is "small".

Several procedures for approximately factoring A have been suggested, see (A7), (D13), (K4), (M16) and (W7). If A is the matrix generated from the application of a 5-point formula for (1.2.2), we find a system with a coefficient matrix A, in the form

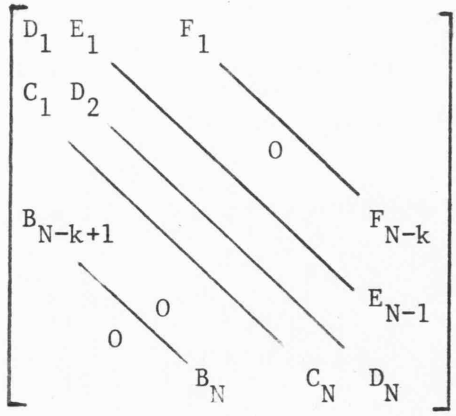


Fig. 1.1

(k is the number of nodes along x-axis)

Define the position of the elements L and U as follows

$$\ell_{ij} = 0 \quad \text{if} \quad a_{ij} = 0 \qquad i > j$$
$$u_{ij} = 0 \quad \text{if} \quad a_{ij} = 0 \qquad j > i \qquad 1.2.7$$

To guarantee the existence of an approximate LU decomposition, L and U are preserved to lose zeros in off diagonal places which are chosen in advance. These places (i,j) can be fixed by the set

$$P \subseteq P_N = \{(i,j) \mid i \neq j, \quad 1 \leq i \leq N, \quad 1 \leq j \leq N\} \qquad 1.2.8$$

where P_N contains all pairs of indices of off-diagonal matrix entries. The various algorithms arise by defining these sets P_N (see A7, B10, D13, D14, G4, J1, J6, K4, M7, M8, M16, S1, S15). Some choices for special matrices will be described in more detail.

The notion of factorization was firstly proposed by Buleev (B24) and Oliphant (O1). Their technique was the basis of Stone's method (S15). This method was essentially the basis of the determination of approximate factorizations for the matrix A, whether A is symmetric or asymmetric. One disadvantage of Stone's factorization was that the product LU = A + B is an asymmetric matrix, even when A is symmetric, the non-symmetricity appearing because of the non-symmetricity of B. To avoid such unsatisfactory behaviour, an improvement was found by A. Brach-Barak and P. Saylor (B10). The main advantage in producing symmetric factorization is to reduce the store requirement by about 40%.

P.E. Saylor, 1974 (S1), determined a class of strongly implicit symmetric factorization methods. Saylor derived this class from the paper of H. Stone (S15) and T. Dupont et al (D14). D.A.H. Jacobs (J1) described a generalization of Stone Factorization to factorize the

matrix generated from the application of 13 points finite difference formula associated with the biharmonic and similar fourth order elliptic equations.

The feature of the matrix factorization has been widely applied in combination with other iterative techniques to solve the system of linear equations, the main advantage of such a combination is to reduce the spectral radius of the iterative matrix. Also, such a combination has been widely applied with conjugate gradient methods (see Chapter 3).

A. Jennings and G.M. Malik, 1977 (J6), proposed a modification of Tuff and Jennings method, 1973 (T3). Meijerink and Van der Vorst, 1977 (M7), proposed a class of regular splittings of any matrix which need not be symmetric M-matrices (for definition of M-matrix see (V1)).

D.S. Kershaw, 1978 (K4), improved the method of Meijerink and Van der Vorst in the case of positive definite symmetric matrices and further generalized it to apply to general non-singular matrices as well.

O. Axelsson, 1979 (A7), described matrix factorization for non-self adjoint problems, and in 1979 K. Meijerink and Van der Vorst (M8) proposed incomplete decomposition for the following kind of matrices

1. Symmetric M-matrices (for definition see (V1))
2. Symmetric positive definite matrices
3. Non-symmetric matrices

1.3 Examples of Matrix Factorization

As has been pointed out before, the elements of the matrices L and U can be determined by using the set P as in 1.2.7 to define the elements in L and U which will be zero. The resulting matrix from the product LU is as follows

LU =

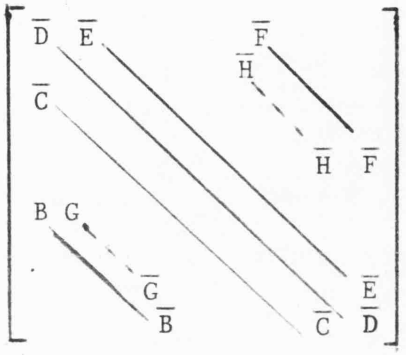


Fig.1.3.1.

If we equate the non-zero elements in the ith row of the resulting matrix LU to the corresponding elements in A we obtain the following relations

$$\bar{B}_i = b_i \qquad N+1 \leq i \leq N \times N$$

$$\bar{G}_i = b_i e_{i-N+1} \qquad N \leq i \leq N \times N$$

$$\bar{C}_i = c_i \qquad 2 \leq i \leq N \times N$$

$$\bar{D}_i = d_i + c_i e_i + b_i f_i \qquad 1 \leq i \leq N \times N \qquad 1.3.1$$

and

$$\bar{E}_i = d_{i-1} e_i \qquad 2 \leq i \leq N \times N$$

$$\bar{H}_i = c_{i-n+1} f_i \qquad N \leq i \leq N \times N$$

$$\bar{F}_i = d_{i-n} f_i \qquad N+1 \leq i \leq N \times N$$

1.3.1 Meijerink and Van der Vorst Factorization (M7)

If we choose

$$\bar{B}_i = B_i, \quad \bar{C}_i = C_i, \quad \bar{D}_i = D_i$$

we will have the following factorization

$$b_i = B_i$$

$$c_i = C_i$$

$$d_i = D_i - C_i e_i - B_i f_i \qquad 1.3.2$$

$$e_{i+1} = E_{i+1}/d_i$$

$$f_{i+n} = F_{i+n}/d_i$$

This computational part of Meijerink and Van der Vorst's incomplete cholesky decomposition, 1977 (M7), is applicable to any kind of matrix; but when A is symmetric such a factorization can be combined with the conjugate gradient method. The resulting preconditioned conjugate gradient method is referred to as ICCG(0).

In general for any matrix real symmetric A, we first compute an approximate matrix factorization LL^T, where L is a lower triangular matrix, to fix the entries which will be forced to be zero in L. Secondly, we compute an approximate factorization LL^T of A by the cholesky algorithm so that L_{ij} is set to zero for $(i,j)\varepsilon P$, (so the elements of L of indices corresponding to the indices in P are neither calculated nor stored).

1.3.2 Dupont, Kendall and Rachford Factorization (see D14)

Dupont et al proposed the following choices for matrix factorization

$$\bar{B}_i = B_i, \quad \bar{C}_i = C_i, \quad \bar{D}_i = D_i(1+\alpha) - \bar{G}_i - \bar{G}_{i+n-1} \qquad 1.3.3$$

for a matrix factorization of A where α is a parameter. By this definition of L and U, we have

$$(LU)u_{ij} = B_{ij}u_{i-nj} + \bar{G}_i(u_{i-n+1} - u_i) + C_i u_{i-1} + D_i(1+\alpha)u_i$$
$$+ E_{i+1}u_{i+1} + \bar{H}_{i+n-1}(u_{i+n-1} - u_i) + F_{i+n}u_{i+n} \qquad 1.3.4$$

and

$$(Au)_{ij} = B_{ij}u_{i-nj} + C_{ij}u_{i-1j} + D_{ij}u_{ij} + E_{i+1j}u_{i+1j} + F_{i+nj}u_{i+nj} \qquad 1.3.5$$

From the above form of factorization, Dupont et al attempt to reduce the effect of the $\overline{G}_i u_{i-n+1}$ term by subtracting $\overline{G}_i u_i$ and the effect of $\overline{H}_{i+n-1}\ u_{i+n-1}$ term by subtracting $\overline{H}_{i+n-1}u_i$ (where $\overline{G}$ and $\overline{H}$ terms are appeared from the multiplications of L and U). The parameter α is used to accelerate the convergence of the algorithm. This is the main difference from the Meijerink and Van der Vorst algorithm which makes no attempt to reduce the effect of these extra elements appearing in the factorization. Therefore we have the following forms of the decomposition LU of the matrix A.

$$\begin{aligned}
b_i &= B_i \\
c_i &= C_i \\
d_i &= D_i(1+\alpha) - b_i e_{i-n+1} - c_i f_{i+n-1} - c_i e_i - b_i f_i \qquad 1.3.6 \\
e_{i+1} &= E_{i+1}/d_i \\
f_{i+n} &= F_{i+n}/d_i
\end{aligned}$$

The evaluation of b_i, c_i, d_i, e_{i+1}, f_{i+n}, can easily be calculated provided that $d_i \neq 0$, $1 \leq i \leq N$.

For all values of α, $0 \leq \alpha \leq 1$, the $\{d_i\}_{i=1}^{N}$ are strictly positive and therefore the resulting matrix LU is positive definite. Dupont (13) proved that if $\alpha = k_o h^2$, where $h = \frac{1}{(n+1)}$ and $k_o \geq 0$ is a constant independent of h, and if the coefficients of the self adjoint elliptic equation are uniformly Lipschitz continuous in each of the directions, x_i, then condition number of the preconditioned matrix is $O(h^{-1})$.

Dupont extended his algorithm, and he proved facrorizability for the nine points discrete Laplacian as well as for the discretized self adjoint Neuman problem.

1.3.3 Stone Matrix Factorization (see S15)

Consider the following self adjoint partial differential equation of second order

$$- \sum_{i=1}^{N} \frac{\partial}{\partial x_i} \left(a_i(x) \frac{\partial u}{\partial x_i}\right) = f(x) \qquad \text{on } D \qquad 1.3.7$$

with $u(x) = g(x)$ for all $x \varepsilon \partial D$.

Where $a_i(x) > 0$, and $f(x)$, $g(x)$ and $a_i(x)$ be sufficiently smooth, let D be a unit square, i.e. $D = \{(x,y) \mid 0 \leq x \leq 1 \text{ and } 0 \leq y \leq 1\}$.

By using a five point finite difference approximation the approximation of $\frac{\partial}{\partial x_i} \left(a_i(x) \frac{\partial u}{\partial x_i}\right)$ is given by

$$- \frac{1}{h_i^2} \left[a_i\left(x + \frac{h_i}{2}\right) \{u(x) - u(x + h_i)\} + a_i\left(x - \frac{h_i}{2}\right) \{u(x) - u(x - h_i)\}\right] \qquad 1.3.8$$

where h_i is a mesh size associated with each mesh point, if we take an equi mesh size, i.e. $h = h_i$ for all i, then 1.3.8 can be simplified to

$$- \frac{1}{h^2} \left[a_i\left(x + \frac{h}{2}\right) \{u(x) - u(x + h)\} + a_i\left(x - \frac{h}{2}\right) \{u(x) - u(x - h)\}\right] \qquad 1.3.9$$

By application of the above approximation 1.3.9 on the Dirichlet problem 1.3.7, we find a system of linear equations $Au = f$, where A has a special structure; it is a five diagonal matrix, where the associated points are of the following scheme (i,j), (i-1,j), (i+1,j),(i,j-1) and (i,j+1).

Thus a single equation is given by

$$[Au]_{ij} = f_{ij} = B_{ij}u_{ij-1} + D_{ij}u_{i-1,j} + E_{ij}u_{ij} + D_{i+1j}u_{i+1j} + B_{ij+1}u_{ij+1} \qquad 1.3.10$$

where $B_{ij} = -a_2(x_1+ih,\ x_2 + (j - \frac{1}{2})h)/h^2$

$D_{ij} = -a_1(x_1 + (i - \frac{1}{2})h,\ x_2 + jh)/h^2$ 1.3.11

$E_{ij} = B_{ij} + D_{ij} + D_{ij+1} + B_{ij+1}$

The idea of Stone is to find an approximate factorization LU for the matrix A such that A + B = LU, the elements of the matrix B are defined explicitly from the elements of the matrix A, and the product of L and U gives the matrix of the following form, where the elements indicated by the dotted line in fig. 1.3.1 correspond to the grids (j + 1, k - 1) and (j - 1, k + 1).

$$\{(A + B)u_{ij}\} = \{(LU)u_{ij}\} = b_{ij}u_{ij} + b_{ij}e_{ij-1}u_{i+1j} + c_{ij}u_{i-1j}$$
$$+ (d_{ij} + b_{ij}f_{ij-1} + c_{ij}e_{i-1j})u_{ij} + d_{ij}e_{ij}u_{i+1j}$$
$$+ c_{ij}f_{i-1j}u_{i-1+j} + d_{ij}f_{ij}u_{ij+1} \qquad 1.3.12$$

Equating LU with A at ij gives the following relations for the unknowns b_{ij}, c_{ij}, d_{ij}, e_{ij} and f_{ij}.

$$b_{ij} = B_{ij}$$
$$b_{ij}e_{ij-1} = 0$$
$$c_{ij} = D_{ij}$$
$$d_{ij} + b_{ij}f_{ij-1} + c_{ij}e_{i-1j} = E_{ij} \qquad 1.3.13$$
$$d_{ij}e_{ij} = D_{i+1j}$$
$$c_{ij}f_{i-1j} = 0$$
$$d_{ij}f_{ij} = B_{ij+1}$$

Stone proposed to determine the value of the listed unknowns by considering the extra diagonals, i.e. $(i + 1, j - 1)$ and $(i - 1, j + 1)$. By using the Taylor expansions of $u_{i+1,j-1}$ and $u_{i-1,j+1}$ about u_{ij} and by subtraction of Taylor expansions of $u_{i,j+1}, u_{i-1,j}, u_{i+1,j}$ and u_{ij-1} we find the following:

$$u_{i+1\ j-1} = -u_{ij} + u_{i+1j} + u_{ij-1} + O(h^2) \qquad 1.3.14$$

$$u_{i-1\ j+1} = -u_{ij} + u_{i-1j} + u_{ij+1} + O(h^2) \qquad 1.3.15$$

Extra flexibility is given to the method by considering "partial cancellation", in which we use a parameter $\alpha \varepsilon [0,1]$.

i.e. $$u_{i+1,j-1} \simeq \alpha(-u_{ij} + u_{i+1,j} + u_{ij-1}) \qquad 1.3.16$$

$$u_{i-1,j+1} \simeq \alpha(-u_{ij} + u_{i-1,j} + u_{ij+1}) \qquad 1.3.17$$

By using the following approach, the i-th equation is given by

$$\begin{aligned} &B_{ij}u_{ij-1} + D_{ij}u_{i-1j} + E_{ij}u_{ij} + D_{i+1j}u_{i+1j} + B_{ij+1}u_{ij+1} \\ &\quad + F_{ij}\left[u_{i+1j-1} - \alpha(-u_{ij} + u_{i+1j} + u_{ij-1})\right] \\ &\quad + G_{ij}\left[u_{i-1j+1} - \alpha(-u_{ij} + u_{i-1j} + u_{ij+1})\right] \end{aligned} \qquad 1.3.18$$

where $F_{ij} = b_{ij}e_{ij-1}$ and $G_{ij} = c_{ij}f_{i-1j}$ and by comparing with $(A + B) = LU$ gives the following relations

$$\begin{aligned} b_{ij} &= B_{ij} - \alpha b_{ij}e_{ij-1} = B_{ij}/(1 + \alpha e_{ij-1}) \\ c_{ij} &= D_{ij} - \alpha c_{ij}f_{i-1j} = D_{ij}/(1 + \alpha f_{i-1j}) \\ d_{ij} &= E_{ij} + b_{ij}(\alpha e_{ij-1} - f_{ij-1}) + c_{ij}(\alpha f_{i-1j} - e_{i-1j}) \\ e_{ij} &= (D_{i+1j} - \alpha b_{ij}e_{ij-1})/d_{ij} \\ f_{ij} &= (B_{ij+1} - \alpha c_{ij}f_{i-1j})/d_{ij} \end{aligned} \qquad 1.3.19$$

With the matrix A + B thus defined, the iterative method is derived by adding Bx to both sides of the linear system Ax = b and Ax - Ax to the right hand side to give

$$(A + B)x = (A + B)x - (Ax - b) \qquad 1.3.20$$

Since A + B is easily factored 1.3.20 provides the basic for an iterative method

$$(A + B)x_{i+1} = (A + B)x_i - (Ax_i - b) \qquad 1.3.21$$

In general values of x for the (i+1)st iterative level can be calculated from x at the i-th level, and the iteration method defined by 1.3.21 is called an S.I.P. method.

1.4 Convergence of the S.I.P. method

To solve any large linear system numerically by the application of any iterative technique, a major problem is to determine how fast iterates will approach the fixed point of the given problem; in other words, the principal part of any such iterative technique τ is the algorithm $G = G(F)$ constituting a single iteration step. We can define the input of G to be $\{i,x,M\}$, where i is an iteration index, x the current iterative vector and M is a set of memory. Therefore the generic iterative process τ is the following scheme.

τ: 1. input $\{x_0, M_0\}$

2. for i = 0,1,...

3. output x_i

4. if x_i is acceptable - STOP

5. Otherwise, $\{x_{i+1}, M_{i+1}\} = G\{i, x_i, M_i\}$ 1.4.1

A stationary one step process G (see Y1) is therefore defined by

$$x_{i+1} = G(x_i) \qquad i = 0,1,\ldots$$

where $G_i : R^n \longrightarrow R^n$ 1.4.2

is the associated iterative operator. For example in Newton's method (see 04), G is given by

$$Gx_{i+1} = x_i - F'(x_i)F(x_i) \qquad 1.4.3$$

The point of attraction of τ is any $x^* \in R^n$ for which there exists an open neighbourhood $S \subseteq D$ (where D is the domain of definition) with the property that if $x_0 \in S$, the sequence $\{x_i\}$ of (1.4.2) remains in D, and converges to x^*.

In the case of affine space we have the following theorem (see 04, R3, V1).

Theorem 1.4.1

For an iteration operator G, if $Gx = Bx + z$, where B is a linear operator, then the sequence of iteration $\{x_i\}$ generated by 1.4.2 converges to the unique fixed point x^* of G in R^n, starting from any $x^* \in R^n$ if and only if B has spectral radius $\zeta(B) < 1$.

The above theorem 1.4.1 gives necessary and sufficient conditions for the existence and uniqueness of solution and the convergence of a starting iterative technique, such as

$$x_{i+1} = x_i - \alpha_{i+1}(Ax_i - b) \qquad \tau = 0,1,2,\ldots \qquad 1.4.4$$

In the method of simultaneous displacement let

$$e_i = x_i - A^{-1}b \qquad 1.4.5$$

Then the error vector can be shown to satisfy the equation

$$e_{i+1} = (I - \alpha A)e_i = (I - \alpha A)^i e_0 \qquad 1.4.6$$

for any fixed α.

The operator $(I - \alpha A)$ is called the "Error Operator", and for such an iterative process the necessary criterion for convergence is that the spectral radius of $(I - \alpha A)$ is less than unity, a sufficient condition for which is that

$$||I - \alpha A|| < 1 \qquad 1.4.7$$

If we assume that the set of eigenvalues of $\{\lambda_i\}_{i=1}^{N}$ of A are bounded by the values of a and b such that

$$0 < a \leq \lambda_1 \leq \lambda_2 \ldots\ldots \leq \lambda_N \leq b < \infty \qquad 1.4.8$$

then the criterion for convergence makes it necessary that

$$|1 - \alpha\lambda_i| < 1 \qquad i = 0,1,\ldots \qquad 1.4.9$$

Therefore the permissible range of values of α is

$$0 < \alpha < \frac{2}{b} \qquad 1.4.10$$

and the best convergence rate is obtained by choosing α so that the spectral radius of $(I - \alpha A)$ is minimized, clearly the best choice of α is that

$$1 - \alpha a = - (1 - \alpha b)$$

Therefore

$$\alpha = \frac{2}{a + b} \qquad 1.4.11$$

The convergence factor for i = 1 (minimum eigenvalue) and i = N (maximum eigenvalue) is

$$|1 - \alpha\lambda_i| \leq \frac{b - a}{b + a} = \frac{P - 1}{P + 1} \qquad 1.4.12$$

where $p = \frac{b}{a}$ is the "P-condition" number of the matrix A.

In general as a matter of comparison between any kind of iterative process generated a sequence of vectors $\{x_i\}$ is interested how quickly the absolute error

$$e_i = ||x_i - x^*||_\infty \qquad \text{for } i = 0,1,... \qquad 1.4.13$$

(where x^* is a solution vector of $Ax = b$), remains below any given tolerance (see R4).

Another question of interest concerns the number of iterations (say n) required to reduce the relative error in the 2-norm below a given tolerance (say ε), so that

$$\left| \frac{1 - \lambda_N/\lambda_1}{1 + \lambda_N/\lambda_1} \right|^n \leq \varepsilon \qquad 1.4.14$$

Such a value of n is given by $n \geq \frac{1}{2} \frac{\lambda_1}{\lambda_N} \ln \frac{1}{\varepsilon}$ (see A6).

But in the case of using a variable parameter, such as $\{\alpha_i\}$, we want to minimize the spectral radius of the corresponding iteration matrix polynomial

$$P_n(A) = \prod_{i=0}^{n} (I - \alpha_i A) \qquad 1.4.15$$

(Note: for a fixed α, we have the iterative matrix $(I - \alpha A)$ and $P_n(A) = (I - \alpha A)^n$).

The error iterative matrix polynomial is given in 1.4.15 so the eigenvalues of $P_n(A)$ are given by

$$P_n(\lambda) = \prod_{i=0}^{n} (1 - \alpha_i \lambda) \qquad 1.4.16$$

where $P_n(0) = 1$.

Let m and M be the extreme eigenvalues of A, then the L_2 norm is majorized by $M_\ell = \max_{m \leq x \leq M} \left| \prod_{i=1}^{\ell} (1 - \tau_i x) \right|$. Since P_n is a polynomial of degree, n, which has a maximum and minimum absolute value on $[m,M]$ we can write it as

$$P_k(x) = T_k\left(\frac{M + m - 2\lambda}{M - m}\right)\left(T_k\left(\frac{m + M}{M - m}\right)^{-1}\right) \qquad 1.4.17$$

where $T_k(\lambda) = \cos(k \arccos(\lambda))$ is the ordinary Tcheydsev polynomial defined over the interval $[-1,1]$, with k-zeros at $\cos((2i+1)\,\pi/2k)$ for $i = 0,1,2,\ldots, k-1$.

Therefore

$$\alpha_i = \frac{2}{M + m + (M - m)\cos((2i + 1)\pi/2k)} \qquad i = 0,1,\ldots,k-1 \qquad 1.4.18$$

Hence to speed up the iteration technique we shall derive a certain matrix C from A such that $\zeta(C^{-1}A) = 1$, i.e. the main purpose of pre-conditioning is to minimize the spectral radius of the associated matrix over an interval $[a,b]$ covering the spectrum of the preconditioned matrix $C^{-1}A$. Many such matrices C exist, for instance there is one for each particular iteration technique, for example in S.O.R. the matrix C is given by

$$C = A + (1 - \omega)E + F, \text{ where } A = D + E + F \qquad 1.4.19$$

As a point of interest we can show that the S.O.R. iterative technique can be formulated in a form essentially equivalent to S.I.P. (see 1.3.21) where the matrix (A + B) has the form of 1.4.19.

The iterative matrix associated with the S.I.P. is $(I - \tau(A + B)^{-1}A)$ by the application of theorem 1.4.1, the necessary and sufficient condition required for the convergence of the iterative technique, we conclude that

$$\zeta(I - \tau(A + B^{-1})A) < 1 \qquad 1.4.20$$

where the generalized S.I.P. to solve the linear system Ax = b is given by

$$(A + B)x_{i+1} = (A + B)x_i - (Ax_i - b))$$

If we assume the eigenvalues of $I - \tau(A + B)^{-1}A$ are bounded by $\bar{a}$ and $\bar{b}$, then

$$0 < \bar{a} \leq \bar{\lambda}_i \leq \bar{b} < \infty \qquad i = 1,2,\ldots,N \qquad 1.4.21$$

Therefore the best choice of τ, to speed up the iterative process is

$$\tau = \frac{2}{\bar{a} + \bar{b}}$$

and we have that

$$|1 - \tau\lambda_i| \leq \frac{\bar{b} - \bar{a}}{\bar{b} + \bar{a}} = \frac{\bar{P} - 1}{\bar{P} + 1}$$

where $\bar{P} = \dfrac{\lambda_{max}}{\lambda_{min}}$

CHAPTER 2 Solving a System of Non-Linear Equations

2.1 Introduction

Let $f:R \longrightarrow R$ be a non-linear operator where R is a space of real variables. To find an approximate solution for the $f(x) = 0$, let x_0 be an approximation of x^*, where $f(x^*) = 0$. Consider the linear polynomial P which we will use to find an approximate solution of x^*, where $f(x^*) = 0$, i.e.

$$P(x) = \alpha(x - x_0) + f(x_0) \qquad 2.1.1$$

Then we have the following iterative scheme

$$x_{k+1} = x_k - \alpha^{-1} f(x_k) , \qquad k = 0,1,... \qquad 2.1.2$$

For a N-dimensional operator F, $F:R^n \longrightarrow R^n$, we have the following

$$P(x) = A(x - x_0) + F(x_0) \qquad 2.1.3$$

(where A is a scalar matrix, $A = \alpha I$).

Simplifying 2.1.3 we get the following relation

$$x_{k+1} = x_k - A^{-1} F(x_k) \qquad k = 0,1,... \qquad 2.1.4$$

2.1.3 can be considered as a generalization of 2.1.1, where 2.1.1 can be considered at each component of the vector x and of the non-linear operator.

It has been found that the best choice of α is the slope $f'(x_0)$ of the tangent f at x_0 (see 04). Similarly in the N-dimensional case the best choice of A is $A = (\frac{\partial F}{\partial x})_{(x_i)}$, $i = 0,1,2,...$, where $(\frac{\partial F}{\partial x})_{(x_i)}$ is the G-derivative at x_i (for definition of G-derivative see (04)).

Therefore at each step of iteration we will have to evaluate $(\frac{\partial F}{\partial x})_{(x_i)}$ at each step i of iteration and 2.1.4 can be written as

$$x_{i+1} = x_i - \left(\frac{\partial F}{\partial x}\right)^{-1}_{(x_i)} F(x_i) , \qquad i = 0,1,... \qquad 2.1.5$$

For 1-dimensional operator 2.1.5 can be expressed as

$$x_{i+1} = x_i - (f'(x_i))^{-1} f(x_i) \qquad 2.1.6$$

The above, of course, is called "Newton's method".
Each step of Newton's method requires the evaluation of $\frac{\partial F}{\partial x}$ which is very time consuming.

One way of reducing the work per step required by Newton's method (which mainly consists of the evaluation of the Jacobian) was proposed by R.P. Brent, 1973 (B22), who used a class of secant methods to avoid the evalaution of the Jacobian at each step of the iteration.

Another technique which has been proposed is to apply a difference approximation to each component $\partial_j F_i$ of $\frac{\partial F}{\partial x}$ as

$$\partial_j F_i(x) = \frac{1}{h_{ij}} (F_i(x + h_{ij} e^j) - F_i(x)) \qquad 2.1.7$$

where h_{ij} is the given discretization parameter, i.e. the step size, and e^j is the j-th coordinate vector.

Let J(x,h) be the difference approximation of $\frac{\partial F}{\partial x}$ at x, where h is a constant mesh size, then the Newton iterative scheme has the following form

$$x_{i+1} = x_i - J(x_i,h_i)^{-1} F(x_i) \qquad 2.1.8$$

2.1.8 is called the "Newton Discretized Iterative Formula" (see 04).
Furthermore we could reduce the amount of work required by revaluating $\frac{\partial F}{\partial x}$ only occasionally. Then the iteration becomes

$$x_{i+1} = x_i - \left(\frac{\partial F}{\partial x}\right)_{p(i)} F(x_i) \qquad i = 0,1,... \qquad 2.1.9$$

where p(i) is some integer less than or equal to i. The limiting cases of 2.1.9 are, of course, p(i) = i, which gives Newton's method, and p(i) = 0, which gives the simplified Newton's method. Now suppose that F' is revaluated every m-step. Then if we renumber the iterates, so that now x_i denotes the im-th iterate of 2.1.9, the iteration is equivalent to

$$x_{i+1} = x_{i,m},\ x_{i,k} = x_{i,k-1} - \left(\frac{\partial F}{\partial x}\right)_i^{-1} F(x_{i,k-1}) \qquad k = 1,\ldots,m$$

$$x_{i,0} = x_i \qquad 2.1.10$$

which, for instance, when m = 2, may be written as

$$x_{i+1} = x_i - \left(\frac{\partial F}{\partial x}\right)^{-1}_{(x_i)} \left\{F(x_i) + F\left(x_i - \left(\frac{\partial F}{\partial x}\right)^{-1}_{x_i} F(x_i)\right)\right\} \qquad i = 0,1,\ldots$$

2.1.11

The iteration 2.1.11 may be considered as the composition of one Newton's step with m - 1 simplified Newton's steps (m = 2). One way of utilizing any kind of iterative method in connection with non-linear equations is as a means of approximating solutions of the linear systems which must be solved to carry out Newton's method, for example we could have a composite Newton-S.O.R. iteration, with Newton's method as the "primary iteration", and S.O.R. as the "secondary iteration". Consider the Newton's iterative method to solve a non-linear operator F.

$$x_{i+1} = x_i - \left(\frac{\partial F}{\partial x}\right)^{-1}_{(x_i)} F(x_i) \qquad 2.1.12$$

Then by decomposing the matrix $J_i = \left(\frac{\partial F}{\partial x}\right)_{(x_i)}$ as

$$J_i = D_i - L_i - U_i \qquad 2.1.13$$

where D_i, L_i and U_i are diagonal strictly lower triangular and upper triangular matrices respectively. Then the one step Newton-S.O.R. iteration is

$$x_{i+1} = x_i - \omega_i (D_i - \omega_i L_i)^{-1} F(x_i), \qquad i = 1,2,... \qquad 2.1.14$$

For more about such a combination see (C6,C7) for Jacobi-Newton's method and Gauss-Seidel-Newton method (see (H8)), and for general analysis and more about generalization of linear iterative method see (03, 04).

In general we can consider all the linear iterative techniques as potential secondary iteration procedures used with Newton's method, as a primary iteration technique.

As is well known the major categories of iteration to solve a system of non-linear equations are

1. Linearization (which requires the evaluation of the non-linear operator and the solution of the resulting linear system by any iterative technique for instance, and then re-evaluating the non-linear operator by using the last iterative solution vector).
2. Non-linearization.

2.2 Mildly non-linear elliptic equations

A non-linear partial differential equation of the general form

$$a\, u_{\tilde{x}\tilde{x}} + 2b\, u_{\tilde{x}\tilde{y}} + c\, u_{\tilde{y}\tilde{y}} = f(\tilde{x},\tilde{y},\, u, u_{\tilde{x}}, u_{\tilde{y}}) \qquad 2.2.1$$

$$(a^2 + b^2 + c^2 \neq 0)$$

defined over the region D, is called "mildly non-linear".

Equation 2.2.1 is elliptic when $b^2 - ac < 0$, and it can be transformed to

$$\alpha\, u_{xx} + \gamma\, u_{yy} = f(x,y,u,u_x,u_y) \quad , \alpha, \gamma > 0 \qquad 2.2.2$$

(for such a transformation see W.F. Ames, 1965 (A4)).

To obtain the numerical solution to (2.2.2) we reduce the continuous infinite dimensional problem to a discrete finite dimensional one. Several techniques are available to do this and we consider the use of finite difference approximations. (Finite element methods could, of course, be used). Finite difference approximations for u_x, u_y, u_{xx} and u_{yy} are:

$$u_x|_{ij} = \frac{1}{2h}(u_{i+1,j} - u_{i-1j}) + O(H^2)$$

$$u_y|_{ij} = \frac{1}{2k}(u_{ij+1} - u_{ij-1}) + O(k^2)$$

$$u_{xx}|_{ij} = \frac{1}{h^2}(u_{i+1j} - 2u_{ij} + u_{i-1j}) + O(h^2)$$

$$u_{yy}|_{ij} = \frac{1}{k^2}(u_{ij+1} - 2u_{ij} + u_{ij-1}) + O(k^2) \qquad 2.2.3$$

where h and k are the mesh size along x and y axes as usual.
Therefore (2.2.3) can be used to approximate equation 2.2.2 as follows

$$\frac{\alpha}{h^2}(u_{i+1j} - 2u_{ij} + u_{i-1j}) + \frac{\gamma}{h^2}(u_{ij+1} - 2u_{ij} + u_{ij-1}) = f_{ij} \qquad 2.2.4$$

For a region such as a square we can use equi-mesh size (i.e. h = k) and then derive

$$\alpha(u_{i+1j} - 2u_{ij} + u_{i-1j}) + \gamma(u_{ij+1} - 2u_{ij} + u_{ij-1}) = h^2 f_{ij} \qquad 2.2.5$$

In general the resulting system of non-linear equation is of the following form

$$Au = F(u) \qquad 2.2.6$$

where A is a matrix of constant coefficients of 5-diagonal entries and F is a non-linear operator.

Now if $\alpha = \gamma = -1$, we have the following problem of the general form

$$\Delta u = f(u) \quad \text{on } D$$

$$\text{with} \quad u(x,y) = g(x,y) \quad \text{on } \partial D \qquad 2.2.7$$

To approximate the above equation 2.2.6, we replace the differential operator by a five point difference approximation on a square grid D_h, with boundary ∂D_h, where h is the mesh size, we find the equations

$$u_{i-1j} + u_{ij-1} + u_{ij+1} + u_{i+1j} - 4u_{ij} = h^2 f(u_{ij}) \quad \text{for } (ih,jh) \in D_h \qquad 2.2.8$$

and

$$u_{ij} = g_{ij} \quad \text{for } (ih,jh) \in \partial D_h$$

We may write 2.2.8 as an N x N system of non-linear equations (where N is the number of grids along x-axis and along y-axis).

2.3 Solving the system Ax = F(x) using S.I.P. techniques

In order to solve the system of non-linear equations obtained by application of finite difference approximation on mildly non-linear elliptic equations, there are two ways of applying S.I.P. techniques to the non-linear systems namely linearization and non-linearization.

2.3.1 Linearization technique

In order to solve non-linear systems by using the S.I.P. technique, the simplest way is to solve the "linearized" system, and then re-evaluate the non-linear operator, by using the last iterative solution vector. But, with respect to the total number of iterations and total work effort, this is much more costly than non-linearization technique (as we will describe in detail in section 2.5).

The general scheme of iteration is as follows

$$(LU)\, x^k_{i+1} = (LU)\, x^k_i - R_{i,k}$$

$$x^k_0 = x^{k-1} \qquad 2.3.1$$

Note: the index i is for steps of inner iterations and k for outer iterations.

Where $R_{i,k}$ is the residual at x_i^k and LU is the approximate factorization for the associated matrix of constant coefficients associated with the given non-linear system.

In order to stop the iterative scheme 2.3.1, the process is continued for $k \geq 1$ and $i \geq 1$, until

$$||x^{k+1} - x^k||_\infty < \varepsilon_1 \qquad 2.3.2$$

and $||x_{i+1}^k - x_i^k||_\infty < \varepsilon_2$

where the values of ε_1 and ε_2 are dependent on the accuracy of the iterative solution vector, and they are not necessary to be equal.

As is obvious ε_2 is chosen to stop the inner iteration but we can replace ε_2 by performing a fixed number of inner iterations and then evaluate the nonlinear operator, which as a matter of fact will reduce the total amount of work required to solve the non-linear system; in this case the scheme of iteration is as follows

$$LU\ x_{i+1}^k = LU\ x_i^k - R_{i(p),k} \qquad 2.3.3$$

where $i \leq i(p)$ and $i(p) = i \bmod p$.

2.3.2 Non linearization technique

The S.I.P. non-linearization technique to solve the non linear system $Ax = F(x)$ (which can be derived from the general Newton technique to solve non linear operator) has the following scheme

$$(A + B)_i\ x_{i+1} = (A + B)_i\ x_i - R_i \qquad 2.3.4$$

(where R_i is the residual at i-th step and $(A + B)_i$ is the approximate factorization of the evaluated Jacobian at x_i).

The calculation of the approximate factorization $(A + B)_i$ requires a good deal of effort, which is nevertheless worthwhile in practice.

To reduce the work required per each iteration we can use the approximate factorization (A + B) of the linearized system of the system of non-linear equations as a fixed value instead of evaluating the Jacobian and its approximate factorization at each step.

The scheme of iteration is as follows

$$(A + B)x_{i+1} = (A + B)x_i - R_i \qquad i = 0,1,2,... \qquad 2.3.5$$

2.4 Convergence

2.4.1 Convergence of S.I.P. linearization technique to solve non linear system

Consider the scheme of iterations to solve the system of non linear equations given by

$$(A + B)x^k_{i+1} = (A + B)x^k_i - (Ax^k_i - f(x^k)) \qquad 2.4.1$$

where k is the index of outer iterations and i is the index of inner iterations).

If we denote the error vector of the outer iteration at level k and level i of the inner iteration by e^k which is given by

$$e^{(k)}_i = x^{(k)}_i - x^{\alpha(k)} \qquad 2.4.2$$

where $x^{\alpha k}$ is the solution vector of $Ax = f(x^k)$, $x^{\alpha k}$ can be expressed as $A^{-1}f(x^k)$, at the level k of outer iteration.

Therefore the error vector satisfies the relation

$$(A + B)e^{k}_{i+1} = (A + B)e^{k}_{i} - Ae^{k}_{i}$$

$$e_{i+1} = e^{k}_{i} - (A + B)^{-1} A\, e^{k}_{i}$$

$$e_{i+1} = (I - (A + B)^{-1} A)^{i}\, e^{k}_{0} \qquad 2.4.3$$

For simplicity we have

$$e_{i+1} = (I - (A + B)^{-1} A)^{i}\, e_{0} \qquad 2.4.4$$

Relation 2.3.10 holds at any level k of outer iteration by using theorem 1.4.1 we conclude that the necessary and sufficient condition for the convergence of the linearized non-linear system at stage k, the modules of the largest eigenvalues of $(I - (A + B)^{-1} A)$ must be less than unity. In other words we conclude that, because we are solving a linear system (the linearized non-linear system at stage k) the behaviour will be similar to the linear system, therefore the criterion for convergence makes it necessary that

$$|1 - \lambda_i| < 1 \qquad 2.4.5$$

which implies that

$$0 < \lambda_i < 2 \qquad 2.4.6$$

Now we consider the convergence of outer iterations. Let v be the solution vector of the original system of non-linear equations. Then by using the formula 2.4.1 we have the following

$$(A + B)e_{i+1} = (A + B)e_i - J_i e_i \qquad 2.4.7$$

$$e_{i+1} = (I - (A + B)^{-1} J_i)e_i = (I - (A + B)^{-1} J_i)^{i}\, e_0 \qquad 2.4.8$$

Let $M = \max_i |J_i|$, (taken over all non-zero elements of J_i)

Therefore

$$e_{i+1} = (I - (A + B)^{-1} J_i)^{i}\, e_o \leq (I - (A + B)^{-1} M)^{i} e_o \qquad 2.4.9$$

Note that for any sequence of iterations $\{x_i\}$, the question of convergence of the iterations to the associated fixed point is a question of how quickly the sequence of the absolute error $\{e_i\}$ $e_i = ||x_i - x^\alpha||$, where x^α is a fixed point, will convergence to a limit zero (see 04).

The above relation 2.4.9 can be considered as

$$e_{i+1} \simeq (I - (A + B)^{-1}M)^i \, e_0 \qquad 2.4.10$$

will converge to a limit zero if $\zeta(I - (A + B)^{-1}M) < 1$

i.e. $|1 - \lambda_i| < 1 \qquad i = 1,2,\ldots,N$

then

$$0 < \lambda_i < 2 \qquad i = 1,2,\ldots,N \qquad 2.4.11$$

Therefore the generated iterative sequence $\{x_i\}$ will converge to the point x^α, where x^α is the solution vector of 2.2.6, if $\zeta(I - (A + B)^{-1}M) < 1$.

2.4.2 Convergence of S.I.P. to solve non-linear system by non-linearization

Consider the system of non-linear equations

$$Au = F(u) \qquad 2.4.12$$

(where A is an N x N matrix generated from the approximation of the non-linear equation and $\tilde{F}$ is a vector of dimension N corresponding to the non-linear operator). Assume that F is monotically increasing operator. Consider now

$$-Au = -F(u) \qquad 2.4.13$$

Let

$$\mathcal{A}u = -Au \quad \text{and} \quad \tilde{F}(u) = -F(u) \qquad 2.4.14$$

Lemma 2.4.1

If $\tilde{F}$ is a monotically decreasing function, then the function $Fu = Au - \tilde{F}(u)$ is monotically increasing, where A is a matrix with positive diagonal entries and $a_{ij} \leq 0$ for $i \neq j$, and A is positive definite.

Proof

Since A is a matrix with positive diagonal entries and non-positive off-diagonal terms.

Therefore $A^{-1} > 0$ (see V1)

Let u, v be any N-dimensional vector, such that $u < v$.

Therefore $\tilde{F}(v) < \tilde{F}(u)$

Then $- \tilde{F}(u) < - \tilde{F}(v)$ and $- A^{-1}\tilde{F}(u) < - A^{-1}\tilde{F}(v)$

$$u - A^{-1}\tilde{F}(u) < v - A^{-1}\tilde{F}(v)$$

Therefore $Au - \tilde{F}(u) < Av - \tilde{F}(v)$

which implies that $Fu < Fv$ i.e. F is monotically increasing operator.

Definition 2.4.1 (see O5)

For an N x N real valued matrices A, B and C, $A = B - C$ is a regular splitting of A if B is invertible and $B^{-1} \geq 0$, $C \geq 0$.

Definition 2.4.2 (see O5)

For an N x N real valued matrices A, B and C, $A = B - C$ is a weak regular splitting of the matrix A, if B is invertible $B^{-1} \geq 0$, $B^{-1}C \geq 0$ and $C^{-1}B \geq 0$.

Definition 2.4.3 (see O5)

Let A and B be an N x N real valued matrix such that $AB \leq I$ and $BA \leq I$, then B is called "a subinverse of A" and vice versa.

Theorem 2.4.1 (see V1)

If $A = B - C$ is a regular splitting of the matrix A and $A^{-1} \geq 0$, then

$$\zeta(B^{-1}C) < 1$$

Corollary 2.4.1 (see O5)

If $A = B - C$ is a weak regular splitting of the matrix A then

$$0 \leq I - B^{-1}A$$

H.B. Keller (K3) proved the existence of an upper and a lower bound for the solution of the non-linear system of $Ax = \tilde{F}(x)$. For an approximate factorization LU of the real valued matrix we have that $LU = A + B$ which implies that $A = LU - B$. In the original system of non-linear equations $Ax = \tilde{F}(x)$. Let LU be an approximate factorization of the matrix A.

$$LUx^{k+1} = LUx^{k} - (Ax^{k} - \tilde{F}(x^{k}))$$

$$LUx^{k+1} = LUx^{k} - Fx^{k}$$

$$x^{k+1} = x^{k} - (A + B)^{-1}Fx^{k}$$

We are going to prove that the S.I.P. iterative scheme is convergent to the solution of $Fx = Ax - \tilde{F}(x)$.

Theorem 2.4.2

Let $F: D \subset R^{n} \longrightarrow R^{n}$ be a non-linear operator, $Fx = 0$, such that $Fx = Ax - \tilde{F}(x)$, $x \in D = [x_0, y_0]$, x_0 and y_0 are the lower and upper bound respectively.

If $Fy - Fx \geq A(y - x)$, where A is a matrix of positive diagonal elements and non positive off-diagonal elements, then the sequence $y_{k+1} = y_k - B^{-1}Fy_k$ is convergent to y^* and $Fy^* = 0$.

Proof

By lemma 2.4.1 we have that $Fx = Ax - \tilde{F}(x)$ is monotically increasing operator.

Let $x \in [x_0, y_0]$

$$x - B^{-1}Fx = y_1 - y_0 + B^{-1}Fy_0 + x - B^{-1}Fx = y_1 + (x - y_0) + B^{-1}(Fy - Fx)$$

$$= y_1 + (x - y_0) - B^{-1}(Fx - Fy_0)$$

$$\lesssim y_1 + (x - y_0) - B^{-1}(A(x - y_0))$$

$$= y_1 + (I - B^{-1}A)(x - y_0) = y_1 - (I - B^{-1}A)(y_0 - x)$$

since $I - B^{-1}A > 0$ and $x < y_0$

Therefore

$$x - B^{-1}Fx \lesssim y_1 - (I - B^{-1}A)(y_0 - x) < y_1$$

which implies

$$x_0 \lesssim x_0 - B^{-1}Fx_0 \lesssim y_1$$

$$Fy_0 - Fy_1 \gtrsim A(y_0 - y_1)$$

$$- Fy_1 \gtrsim - Fy_0 + A(y_0 - y_1)$$

$$Fy_1 \lesssim Fy_0 - A(y_0 - y_1)$$

$$= Fy_0 + A(y_1 - y_0)$$

Because of $AB^{-1} > 0$ and $Fy_0 > 0$

$$Fy_1 \lesssim Fy_0 + A(- B^{-1}Fy_0) = Fy_0 - AB^{-1}Fy_0 \lesssim Fy_0$$

Therefore $Fy_1 \lesssim Fy_0$

Also we conclude that $Fy_1 > 0$, when $x_0, y_0 > 0$.

Therefore we have a sequence of positive vectors, $Fy_i > 0$, $\{Fy_i\}$ generated from the applications of the monotically decreasing operator F on the sequence $\{y_i\}$ generated from the iteration

$$y_{i+1} = y_i - B^{-1}Fy_i$$

which is bounded below by x_0 and bounded above by y_0

$$x_0 \leq y_i \leq y_0$$

Hence the sequence $\{y_i\}$ generated from

$$y_{i+1} = y_i - B^{-1}Fy_i$$

is monotically decreasing sequence i.e. for each i, $y_{i+1} \leq y_i$.

Therefore $Fy_{i+1} - Fy_i = Ay_{i+1} - \tilde{F}(y_{i+1}) - (Ay_i - \tilde{F}(y_i))$

$$= A(y_{i+1} - y_i) - \tilde{F}(y_{i+1}) + \tilde{F}(y_i) > 0$$

Therefore $Fy_{i+1} > Fy_i$ (contradiction)

Hence $\exists y^*$, which is the convergence point of the sequence $\{y_i\}$

i.e. $\lim_{i\to\infty} y_i = y^*$

$$\lim_{i\to\infty} (y_{i+1} - y_i + B^{-1}Fy_i) = B^{-1}Fy^* = 0$$

Therefore $Fy^* = 0$

CHAPTER 3 Conjugate Gradient Method

3.1 Introduction

In this section we are going to investigate a certain class of variational methods called "Conjugate Gradient", to solve a system of non-linear equations.

It has been mentioned before that there are two main categories of method to consider:

1. Linearization
2. Non-Linearization

The linearization technique includes all iterative methods to solve linear systems as inner iterations.

One well known class of method for linear systems is known as the "Conjugate Gradient" method. for the solution of $Ax = b$ where A is positive definite and symmetric.
This was first proposed by Hestenes and Stiefel in 1952 (H12), as a direct method, which would in the absence of round off error, yield an exact solution of an N^2-system at most N^2-iterative steps. But even for small system (for example $N^2 \lesssim 100$), round off error can seriously contaminate the approximations. Therefore the conjugate gradient method never became popular as a direct method.

Reid (R1) in 1971 indicated that the conjugate gradient method can be very accurate and fast for several problems even though rounding errors cause it to depart significantly from its ideal path, in fact conjugate gradient methods can be relied upon to converge ultimately and so are effective if regarded as an iterative, rather than a direct method. Reid also proved that the conjugate gradient method compared favourably

with methods like Tchebyshev acceleration and S.O.R. for large sparse systems.

Attractive features of conjugate gradient methods are that no further special properties are needed for A, no acceleration parameters have to be estimated and only three of four vectors need to be held in the main store in addition to the demands of the operator A. A useful feature is that the effect of round off error on actual implementation does not prevent convergence but merely delays it. Subsequently, the conjugate gradient method has been recognized as a powerful iterative technique for solving large sparse linear system of equations.

In addition to the 1971 paper by J.K. Reid, in 1972 Reid (R2) has shown that if the coefficient matrix is two cyclic (i.e. a matrix possessing "property A", see (V2) or (Y1)) then the work required for the conjugate gradient method may be approximately halved with also a small saving in the storage required.

3.2 Variational iterative methods

Consider the system of linear equations

$$Ax = b \qquad 3.2.1$$

where A is an N x N real symmetric matrix. For a positive integer μ the functional

$$E_{\mu}(x^{\alpha}) = (x - x^{\alpha}, A^{\mu}(x - x^{\alpha})) \qquad 3.2.2$$

is called the "Error Functional". This is the square of the A^{μ}-norm of $(x - x^{\alpha})$ and it attains its unique minimum when $x^{\alpha} = x$, therefore solving (3.2.1) is equivalent to minimizing (3.2.2).

Different values of μ yield different methods all belonging to the same class (see R. Chandra, 1978, C2).

To approximate x, these methods compute a sequence of iterates $x_0, x_1, x_2, \ldots$ by moving along a sequence of directions $p_0, p_1, p_2, \ldots$ minimizing $E_\mu(x^*)$ along each successive direction.

More precisely, at the i-th step, x_{i+1} is computed as $x_{i+1} = x_i + a_i p_i$, where a_i is chosen to minimize $E_\mu(x_{i+1})$ and the direction vectors are chosen by a method based on the Lanczos algorithm (L1).

By this special way of choosing the direction vectors, the unidirectional minimization corresponds to minimization in the whole subspace generated by the $\{P_i\}$ so that the x_{i+1} obtained actually minimizes the error functional on the affine subspace $x_0 + \{P_0, P_1, \ldots, P_i\}$.

Algorithm 3.2.1

The algorithm of Lanczos method

1) Define $p_{-1} = 0$

   Choose p_0 (an initial approximation)

   Set $i = 0$

2) Iteration i:

   $$\delta_i = (Ap_i, A^\mu p_{i-1})/(p_{i-1}, A^\mu p_{i-1})$$

   $$\gamma_i = (Ap_i, A^\mu p_i)/(p_i, A^\mu p_i)$$

   and

   $$p_{i+1} = Ap_i - \gamma_i p_i - \delta_i p_{i-1}$$

Theorem 3.2.1

The vectors $\{P_i\}$ generated by Lanczos method are A^μ - orthogonal

Proof (see C2)

Using algorithm 3.2.1 for generating the direction vectors, we now consider the following algorithm for computing the approximations x_i to x.

Algorithm 3.2.2

The variational method

1) Choose an initial approximation x_0 to x

   Compute $r_0 = b - Ax_0$

   $p_0 = r_0$

   $i = 0$

2) $a_i = (r_i, A^{\mu-1} p_i)/(p_i, A^\mu p_i)$

   $x_{i+1} = x_i + a_i p_i$

   $r_{i+1} = r_i - a_i A p_i$

   $\delta_i = (Ap_i, A^\mu p_{i-1})/(p_{i-1}, A^\mu p_{i-1})$

   $\gamma_i = (Ap_i, A^\mu p_i)/(p_i, A^\mu p_i)$

   and

   $p_{i+1} = Ap_i - \gamma_i p_i - \delta_i p_{i-1}$

3) If convergence is achieved, STOP

   otherwise set $i = i + 1$, go to 2.

Theorem 3.2.2

For the above variational method, the following relations hold

1) $Ap_i \in \{p_0, p_1, \ldots, p_{i+1}\}$

2) $r_i \in \{p_0, p_1, \ldots, p_i\}$

3) $\{p_0, p_1, \ldots, p_i\} = \{p_0, Ap_0, \ldots, A^i p_0\} = \{r_0, Ar_0, \ldots, A^i r_0\}$

4) $(r_i, A^{\mu-1} p_j) = 0 \qquad j < i$

5) $(r_i, A^{\mu-1} r_j) = 0 \qquad i \neq j$

6) $(r_i, A^{\mu-1} p_j) = (r_0, A^{\mu-1} p_j) \qquad i \leq j$

7) If $r_i \neq 0$, then $p_i \neq 0$

Proof: see (C2)

Theorem 3.2.3

For each i, the iterate x_{i+1} minimizes $E_\mu(x^*)$ over the affine subspace $x_0 + \{p_0, p_1, \ldots, p_i\}$.

Proof:

Let $x^* = x_0 + \sum_{j=0}^{i} t_j p_j$, where $\{t_j\}_{j=0}^{i}$ are scalars.

Then $E_\mu(x^*) = (x - x_0 - \sum_{j=0}^{i} t_j p_j,\ A^\mu(x - x_0 - \sum_{j=0}^{i} t_j p_j))$

$$= (r_0 - \sum_{j=0}^{i} t_j Ap_j,\ A^{\mu-2}(r_0 - \sum_{j=0}^{i} t_j Ap_j))$$

Since the p's are A^μ-orthogonal

$$E_\mu(x^*) = E(x_0) - \sum_{j=0}^{i} (2t_j(Ap_j, A^{\mu-2} r_0) - t_j^2 (p_j, A^\mu p_j))$$

The necessary and sufficient conditions for a minimum are that

$$(Ap_j, A^{\mu-2} r_0) - t_j (p_j, A^\mu p_j) = 0 \qquad j = 0, 1, \ldots, i$$

Hence by theorem 2, we have that

$$t_j = (r_0, A^{\mu-1}p_j)/(p_j, A^{\mu}p_j) = (r_j, A^{\mu-1}p_j)/(p_j, A^{\mu}p_j) = a_j$$

Therefore

$$x^* = x_0 + \sum_{j=0}^{i} a_j p_j = x_{i+1}$$

3.3 Estimated error bounds

From the above variational algorithms, at i-th step the iterative solution vector is

$$x_i = x_{i-1} + a_{i-1}p_{i-1}$$

and, by theorem 3.2.3,

$$x_i = x_0 + \sum_{j=0}^{i-1} t_j A^j r_0$$

$$= x_0 + P_{i-1}(A)r_0$$

where $\{t_j\}_{j=0}^{i=1}$ are certain scalars and $P_{i-1}(A)$ is a polynomial of degree at most i - 1 in A.

Now to obtain the error bounds for our variational method

$$x - x_i = x - x_0 - P_{i-1}(A)r_0$$

$$r_i = r_0 - AP_{i-1}(A)r_0 = (I - AP_{i-1}(A))r_0$$

$$E_\mu(x_i) = (r_i, A^{\mu-2}r_i)$$

$$= (R_i(A)r_0, A^{\mu-2}R_i(A)r_0)$$

where $R_i(A) = I - AP_{i-1}(A)$

$R'_m = \{R_m(y)\}$ $R_m(y)$ is a polynomial of degree at most m in y, $\{R_m(0) = 1\}$

$$E_\mu(x_i) = \min_{R_i \in R'_i} (R_i(A)r_0, A^{\mu-2}R_i(A)r_0)$$

Since A is symmetric, it has N-orthonormal eigenvectors $\{v_j \mid 1 \leq j \leq N\}$

i.e.

$$Av_j = \lambda_j v_j \qquad j = 1,2,\ldots,N$$

where $\{\lambda_j\}_{j=1}^{N}$ are the eigenvalues of A

Since $r_0 = \sum_{j=1}^{N} t_j v_j$ for some scalars $\{t_j\}_{j=1}^{N}$

we have $R_i(A)r_0 = \sum_{j=1}^{N} t_j R_i(A)v_j = \sum_{j=1}^{N} t_j R_i(\lambda_j)v_j$

Therefore

$$E_\mu(x_i) = \min_{R_i \in R_i'} \left(\sum_{j=1}^{N} t_j R_i(\lambda_j)v_j,\ A^{\mu-2} \sum_{j=1}^{N} t_j R_i(\lambda_j)v_j \right)$$

$$= \min_{R_i \in R_i'} \left(\sum_{j=1}^{N} t_j R_i(\lambda_j)v_j,\ \sum_{j=1}^{N} t_j R_i(\lambda_j)\lambda_j^{\mu-2}v_j \right)$$

$$= \min_{R_i \in R_i'} \sum_{j=1}^{N} t_j^2 \ R_i^2(\lambda_j)\lambda_j^{\mu-2}$$

$$\leq \min_{R_i \in R_i'} \left(\max_{1\leq j\leq N} |R_i(\lambda_j)| \right)^2 \left(\sum_{j=1}^{N} t_j^2 \lambda_j^{\mu-2} \right)$$

$$\leq \min_{R_i \in R_i'} \left(\max_{1\leq j\leq N} |R_i(\lambda_j)| \right)^2 \left(\sum_{j=1}^{N} t_j v_j,\ \sum_{j=1}^{N} t_j \lambda_j^{\mu-2} v_j \right)$$

$$= \min_{R_i \in R_i'} \left(\max_{1\leq j\leq N} |R_i(\lambda_j)| \right)^2 \left(\sum_{j=1}^{N} t_j v_j,\ A^{\mu-2} \sum_{j=1}^{N} t_j v_j \right)$$

$$= \min_{R_i \in R_i'} \left(\max_{1\leq j\leq N} |R_i(\lambda_j)| \right)^2 (r_0,\ A^{\mu-2} r_0)$$

$$= Q_i^2 \, E_\mu(x_0)$$

where

$$Q_i = \min_{R_i \in R_i'} \max_{1\leq j\leq N} |R_i(\lambda_j)|$$

Hence we have the following theorem

Theorem 3.3.1

At the i-th iterations step of the variational method satisfy

$$||x - x_i||_{A^\mu} \leq Q_i \; ||x - x_0||_{A^\mu}$$

where Q_i is as has been defined.

Corollary 3.3.1

If r_o has non-zero components along only $k \leq N$ eigenvectors of A corresponding to only $m \leq k$ distinct eigenvalues, then the methods converge in at most m iterations to the unique solution of $Ax = f$.

3.4 Conjugate gradient method

To solve the linear system

$$Ax = b \qquad 3.4.1$$

where A is an N x N symmetric positive definite matrix. In section 2 we studied a certain class of variational method for solving symmetric linear systems. Now we are going to consider the case when the associated matrix is positive definite as well as symmetric.

In section 2 we defined a certain kind of polynomial, $R_i(\lambda)$ of degree i of least deviation from zero on the eigenvalue spectrum of A, such that $R_i(0) = 1$. For general eigenvalue distributions we cannot exactly find the minimum polynomial, and cannot evaluate Q_i, but we can obtain upper bounds for Q_i.

For a positive definite A, let $[a,b]$, $0 < a < b$, be an interval known to contain all the eigenvalues of A, choose $R_i, R_i(\lambda)$ to be the unique polynomial in $R_i'(\lambda)$ that minimizes the deviation from zero on $[a,b]$. Therefore the normalized Tchybshev polynomial on $[a,b]$

$$R_i(\lambda) = \frac{T_i\left(\frac{-2\lambda + a + b}{b - a}\right)}{T_i\left(\frac{b + a}{b - a}\right)}$$

where $T_i(z) = \cos(i \arccos z)$ is the i-th Tchybshev polynomial in z.

Therefore $Q_i \leq 2\left(\frac{1 - \sqrt{\alpha}}{1 + \sqrt{\alpha}}\right)^i$ for $i \geq 0$

where $\alpha = a/b$. Noting that $k(A) = b/a$, therefore we have the following theorem.

Theorem 3.4.1 (see C2)

The iterates x_i, $i \geq 0$, satisfy

$$||x - x_i||_{A^\mu} \leq 2\left(\frac{1 - \sqrt{\alpha}}{1 + \sqrt{\alpha}}\right)^i ||x - x_0||_{A^\mu}$$

where $\alpha = 1/k(A)$.

Axelsson (A6) obtains bounds on Q_i by assuming that the eigenvalues of A are distributed over two intervals on the positive real axis.

Now we are going to consider the alternative form of the variational method to solve the system of linear equations, when A is asymmetric and positive definite system.

Algorithm 3.4.1 (see C2)

The variational method for positive definite coefficient matrices

1. Choose an initial approximation x_0 to x

   $r_0 = b - Ax_0$

   $p_0 = r_0$

   $i = 0$

2. Iteration i

   $a_i = (r_i, A^{\mu-1} r_i)/(p_i, A^\mu p_i)$ or $(a_i = (r_i, A^{\mu-1} p_i)/(p_i, A^\mu p_i))$

   $x_{i+1} = x_i + a_i p_i$

$r_{i+1} = r_i - a_i A p_i$

$b_i = (r_{i+1}, A^{\mu-1} r_{i+1})/(r_i, A^{\mu-1} r_i)$

and

$p_{i+1} = r_{i+1} + b_i p_i$

3. If convergence is achieved (STOP)

   otherwise set i = i + 1 and go to 2.

The conjugate gradient technique can now be derived form the above algorithm by setting $\mu = 1$. Here we present the algorithm as proposed by Reid (R1).

Algorithm 3.4.2

Conjugate Gradient Algorithm

1. Choose x_0, an initial approximation

   $r_0 = b - Ax_0$

   $p_0 = r_0$

   i = 0

2. Iteration i

   $a_i = (p_i, r_i)/(p_i, Ap_i)$

   or

   $= (r_i, r_i)/(p_i, Ap_i)$

   $x_{i+1} = x_i + a_i p_i$

   $r_{i+1} = r_i - a_i Ap_i$

   or

   $= b - Ax_{i+1}$

   $b_i = (-r_{i+1}, Ap_i)/(p_i, Ap_i)$

   or

   $= (r_{i+1}, r_{i+1})/(r_i, r_i)$

$$p_{i+1} = r_{i+1} + b_i p_i$$

3. If r_i or p_i are below any certain tolerance (STOP) otherwise set i = i + 1, and go to 2.

In the above algorithm the different choices of a_i, r_i and b_i are equivalent (for proof see (R1)). The inner product $(x,y) = X^T A^\mu y$, $\mu = 0,1,2,\ldots$, as has been mentioned before will produce different algorithms for different values of μ (see R1). All the previous theorems for estimating the error bounds are valid when we solve a linear system by conjugate gradient method and the related techniques.

3.5 Preconditioning

Consider the linear system

$$Ax = b \qquad 3.5.1$$

where A is an N x N symmetric and positive definite matrix. For any non-singular Q, we could scale the linear system (3.5.1) and solve instead the equivalent linear system

$$A'x' = b'$$

where $A' = Q^{-1} A Q^{-T}$ $\quad b' = Q^{-1} b$ $\qquad 3.5.2$

and $x' = Q^T x$

It is clear that the resulting matrix A' is symmetric and positive definite.

As has been shown before, for the linear system, a lower bound on the rate of convergence is a monotone decreasing function of k(A), and, as in S.I.P., one way to (possibly) improve the rate of convergence is to choose Q to decrease the condition number of the iteration matrix. More precisely Q could be chosen so that

$$k(A') < k(A) \qquad 3.5.3$$

Then by using the previous algorithm, the conjugate gradient method will hopefully converge faster for the preconditioned problem 3.5.2 than for 3.5.1. Since the matrix Q is non-singular, $M = QQ^T$ is symmetric positive definite, conversely any symmetric positive definite matrix M can be written as a product of QQ^T, where Q is non-singular (see (Y1)). Thus the question of choosing an appropriate non-singular Q for scaling the associated matrix A, is equivalent to the question of choosing a splitting of the matrix A of the form $A = C - R$, where C is an N x N symmetric positive definite matrix, such that different splittings correspond to different preconditionings which will produce different rate of convergence according to the relation $\zeta(C^{-1}A) = 1$, where C is a certain approximation for the matrix A, which can easily be inverted, and the best choice of C is $C = A$.

One point to consider, before applying any preconditioning, is that for the preconditioning to be effective, the additional cost of applying it must be sufficiently small.

If for some reason the resulting matrix C does not have the same sparsity pattern as A, then we may need more storage for C and more effort in producing "matrix by vector" products and the overall work per iteration may go up, possibly, substantially. In that case the saving in the total number of iterations for convergence may possibly not compensate for the increased storage and work requirements.

As we will see later the only additional work, to the original technique, is in the solution of the system

$$Cu = v \qquad 3.5.4$$

where C is an approximation for the matrix A and u is the required vector.

As was pointed out before, the resulting matrix C may not necessarily have the same sparsity pattern as the matrix A. In order to avoid fill in, during the factorization of large sparse matrix problems, it is well known that we can use incomplete factorization and accelerate the method by using such a preconditioning (see A6, A7, T3 and M7).

In 1973, D. Jacobs (J1) proposed an S.I.P. technique to factorize a matrix generated from the application of 13-point finite difference approximation to approximate a fourth order partial differential equation and a comparison was made by Jacobs to show the efficiency of the S.I.P. technique.

In 1979 Y.S. Wong (W7) defined a certain factorization to solve a fourth order elliptic partial differential equations by preconditioned conjugate gradients.

Any kind of matrix factorization (for a symmetric system) can be combined with conjugate gradients to produce a more rapid convergence, the rate of convergence depending on how close the matrix $C^{-1}A$ to the identity matrix I, and the efficiency of the technique essentially determined by how easy it is to solve $Cu = v$. As has been remarked by Axelsson (A8), "An infinity of choices of M exists" (M means a matrix factorization of a certain iterative matrix A). We are going briefly to consider some of them.

3.5.1 The generalized SSOR method (see A5)

Let $A = D + L + U$, where D is (block) diagonal, L and U are the (block) triangular lower and upper parts of A, the matrix C derived from A is $C = (\overset{*}{D} + L)\overset{*-1}{D}(\overset{*}{D} + U)$, where D is a suitably chosen diagonal matrix, $\overset{*}{D} = \frac{1}{\omega} D$, $0 < \omega < 2$, i.e. we have the classical block SSOR method.

The advantage of this choice is that one does not have to construct any new entries apart from those in D^*.

3.5.2 Incomplete factorization (see M7)

A method which avoids fill in during factorization is described in a general context by Meijerink and Van der Vorst (M7). In this technique fill in entries are neglected during the factorization. Only fill in of certain positions (chosen in advance) is accepted, so the extent of fill is controlled. (See chapter 1).

As is pointed out by Axelsson (A7), by using such a method the number of operations per unknown in the ICCG method grows as $O(h^{-1})$, $h \to \infty$, in second order positive definite problems. The only problem with incomplete factorization is that the resulting factorized matrix may be unstable since the pivot entries may become negative even if A is positive definite, but Meijerink and V. der Vorst (M7) proved that their factorization is always stable for positive definite and self adjoint problems. A certain remedy has been proposed by Axelsson (A7) to remove the instability in some other cases.

A modification to the incomplete factorization and dynamic fill in has been taken by Munksgard(M16) and Axelsson (A7), that in order to get an arbitrarily accurate incomplete factorization one may allow for dynamic storage of accepted fill in entries, that is, one does not have to prescribe the locations in advance in addition one may also combine the two methods, that is, allow for fill in certain positions and in addition allow fill in entries which are above a certain tolerance or tolerances.

3.6 The preconditioned variational method for positive definite system

Let M be an approximate preconditioning for the matrix A, the algorithm to solve the linear system

$$Ax = b \qquad 3.6.1$$

where A is an N x N matrix, by preconditioned variational method to solve 2.4.5 is as

Algorithm 3.6.1 (C2)

1. Choose x_0

   Compute $r_0 = b - Ax_0$

   Solve $Mz_0 = r_0$

   $p_0 = r_0$

   $i = 0$

2. Iteration i

   $a_i = (r_i(M^{-1}A)^{\mu-1}z_i)/(p_i, A(M^{-1}A)^{\mu-1}p_i)$

   $x_{i+1} = x_i + a_i p_i$

   $r_{i+1} = r_i - a_i Ap_i$

   Solve $Mz_{i+1} = r_{i+1}$ (or $z_{i+1} = z_i - a_i M^{-1}Ap_i$)

   $b_i = (r_{i+1}, (M^{-1}A)^{\mu-1}z_{i+1})/(r_i, (M^{-1}A)^{\mu-1}z_i)$

   and

   $p_{i+1} = z_{i+1} + b_i p_i$

3. If convergence is achieved (STOP)

   otherwise set i = i + 1, go to 2.

Theorem 3.6.1 (C2)

The iterates x_i, $i \geq 0$, of the preconditioned variational method satisfy

$$||x - x_i||_{A(M^{-1}A)^{\mu-1}} \leq 2\left(\frac{1 - \sqrt{\alpha}}{1 + \sqrt{\alpha}}\right)^i ||x - x_0||_{A(M^{-1}A)^{\mu-1}}$$

where $\alpha = 1/k\ (Q^{-1}AQ^{-T})$.

At the i-th step of iteration, the iterates x_i minimizes the $A(M^{-1}A)^{\mu-1}$ norm of the error among all approximations of the form

$$x_i = x_0 + P_{i-1}(M^{-1}A)M^{-1}A(x - x_0)$$

There are several choices for splitting the matrix A into $A = M - R$, in association this kind of splitting with the linear stationary iterative method

$$Mx_{i+1} = Rx_i + b$$

or

$$x_{i+1} = M^{-1}Rx_i + M^{-1}b \qquad 3.6.2$$

simply we can prove that

$$x_i = x_0 + P_{i-1}(M^{-1}A)M^{-1}A(x - x_0)$$

The necessary and sufficient condition for the convergence of the stationary iterative technique that $\zeta(M^{-1}R) < 1$ and $k(Q^{-1}AQ^{-T}) < k(A)$ where $M = QQ^T$.

Theorem 3.6.2 (C2)

If $\zeta(M^{-1}R) < 1$, where $A = M - R$, then the iterates x_i of the preconditioned method satisfy

$$||x - x_i||_{A(M^{-1}A)^{\mu-1}} \leq 2\left(1 - \frac{2\sqrt{1 - \zeta(M^{-1}R)}}{\sqrt{1 + \zeta(M^{-1}R)} + \sqrt{1 - \zeta(M^{-1}R)}^{\,i}}\right).$$

$$||x_0 - x||_{A(M^{-1}A)^{\mu-1}}$$

3.6.1 The preconditioned conjugate gradient method

The preconditioned conjugate gradient is a resulting method from the preconditioned variational method takes on its form when $\mu = 1$.

Algorithm 3.6.2 (C2)

The preconditioned conjugate gradient algorithm

1. Choose x_0

   Solve $Mz_0 = r_0$

   $p_0 = z_0$

   $i = 0$

2. Iteration i

   $a_i = (r_i, z_i)/(p_i, Ap_i)$

   $x_{i+1} = x_i + a_i p_i$

   $r_{i+1} = r_i - a_i Ap_i$

   Solve $Mz_{i+1} = r_{i+1}$

   $b_i = (r_{i+1}, z_{i+1})/(r_i, z_i)$

   and

   $p_{i+1} = z_{i+1} + b_i p_i$

3. If the convergence is achieved, STOP

   otherwise set $i = i + 1$, and go to 2

3.7 Non linear variational iterative methods

In solving the non linear operator $F:R^n \longrightarrow R^n$, $F(x) = 0$, by an iterative method, we try to find an approximate solution $\overset{*}{x}$ by interpreting the error functional, and to be in conformity with the linear $Ax = b$. Thus the error functional according to non linear system is as follows

$$E_\mu(\overset{*}{x}) = (x - \overset{*}{x}, J^\mu_{\overset{*}{x}}(x - \overset{*}{x})) \qquad 3.7.1$$

(where μ is a positive integer constant, and we restrict μ to be even when the Jacobian $J_{\overset{*}{x}}$ is indefinite).

The method minimizes $E_\mu(\overset{*}{x})$ over a subspace of increasing dimension while at the i-th step x_{i+1} is computed as $x_{i+1} = x_i + a_i p_i$, where a_i is chosen to minimize the $E_\mu(x_{i+1})$, and the direction vectors are chosen by a method based on the Lanczos method for generation of orthogonal vectors.

Algorithm 3.7.1

The variational method for non-linear operator $F(x) = 0$, the associated Jacobian $J_i = J(x_i)$ at the i-th steep of iteration.

1. Choose an initial approximation x_0 to x

   $r_0 = F(x_0)$

   $p_0 = r_0$

   $i = 0$

2. Iteration i

   $a_i = (r_i, J_i^{\mu-1} p_i)/(p_i, J_i^\mu p_i)$

   $x_{i+1} = x_i + a_i p_i$

   $r_{i+1} \simeq r_i + a_i J_i p_i \quad (r_{i+1} = F(x_{i+1}))$

   $\delta_i = (J_i p_i, J_i^\mu p_{i-1})/(p_{i-1}, J_i^\mu p_i)$

   $\gamma_i = (J_i p_i, J_i^\mu p_i)/(p_i, J_i^\mu p_i)$

   and

   $p_{i+1} = J_i p_i - \gamma_i p_i - \delta_i p_{i-1}$

3. If convergence is achieved (STOP)

   otherwise set $i = i + 1$ and go to 2.

Theorem 3.7.1

The following relations hold for the variational method for non-linear operator (Algorithm 3.7.1).

1. $J_i p_i \in \{J_0 p_0, J_1 p_1, \ldots, J_{i+1} p_{i+1}\}$

2. $r_i \in \{J_0 p_0, J_1 p_1, \ldots, J_i p_i\}$

3. $\{J_0 p_0, J_1 p_1, \ldots, J_i p_i\} = \{J_0 p_0, J_1 J_0 p_0, \ldots, (\prod_{k=0}^{i} J_k) p_0\}$

$$= \{J_0 r_0, J_1 J_0 r_0, \ldots, (\prod_{k=0}^{i} J_k) r_0\}$$

4. $(r_i, J_j^{\mu-1} p_j) = 0 \qquad j < i$

5. $(r_i, J_j^{\mu-1} p_j) = (r_0, J_j^{\mu-1} p_j) \qquad i \leq j$

6. If $r_i \neq 0$, then $p_i \neq 0$

Proof

By algorithm (3.7.1), (1) follows directly

to prove 2, 3 and 4 we proved by induction

since $p_0 = r_0$, therefore 2, 3 and 4 hold for $i = 0$

Assume for 2, 3 and 4 they hold for $i \leq k$

Now consider $r_{k+1} \simeq r_k + a_k J_k p_k$ and $J_k p_k \in \{p_0, p_1, \ldots, p_{k+1}\}$

therefore we have that $r_{k+1} \in \{p_0, p_1, \ldots, p_{k+1}\}$

thus 2 holds for $i = k + 1$

Now to prove 3

we have that $p_{k+1} \in \{J_0 p_0, J_1 J_0 p_0, \ldots, \prod_{m=0}^{k+1} J_m p_0\}$

therefore we have that $\{p_0, p_1, \ldots, p_{i+1}\} \subseteq \{J_0 p_0, J_1 J_0 p_0, \ldots, \prod_{m=0}^{k+1} J_m p_0\}$

and because the p's are J^{μ}-orthogonal

hence $\{p_0, p_1, \ldots, p_{k+1}\} = \{J_0 p_0, J_1 J_0 p_0, \ldots, \prod_{m=0}^{k+1} J_m p_0\}$

and $\{J_0 r_0, J_1 J_0 r_0, \ldots, \prod_{m=0}^{m=k+1} J_m r_0\} = \{J_0 p_0, J_1 J_0 p_0, \ldots, \prod_{m=0}^{m=k+1} J_m p_0\}$

To prove 4

$$(r_{k+1}, J_j^{\mu-1} p_j) = (r_k, J_j^{\mu-1} p_j) - a_k (J_k p_k, J_j^{\mu-1} p_j)$$

If $j = k$, then $(r_{k+1}, J_j^{\mu-1} p_j) = 0$, by using the definition of a_k, therefore for $j < k$ we have that $(r_i, J_j^{\mu-1} p_j) = 0$ for $j < i$

To prove 5

Since $r_k = r_0 + \sum_{i=0}^{k-1} a_i J_i p_i$

By the orthogonality of $J_i p_i$ to $J_j^{\mu-1} p_j$, therefore 5 holds

To prove 6, assume $r_i \neq 0$ and $p_i = 0$

since $r_i \in \{p_0, p_1, \ldots, p_{i-1}\}$ and

$$\prod_{k=0}^{i} J_k r_i \in \{J_0 p_0, J_1 J_0 p_0, \ldots, \prod_{k=0}^{i-1} J_k p_0\} \subseteq \{p_0, p_1, \ldots, p_{i-1}\}$$

Therefore $(J_i^{\mu} r_i, r_i) = (J_i^{\mu-1} r_i, J_i r_i) = 0$ which implies $r_i = 0$ C!

Corollary 3.7.1

At the i-th step of the iteration we have

$$\{J_0 p_0, J_1 J_0 p_0, \ldots, \prod_{k=0}^{k=i+1} J_k p_0\} \subseteq \{J_0 p_0, \ldots, J_0^{i+1} p_0\}$$

Theorem 3.7.2

For each i, x_{i+1} minimizes $E_\mu(\overset{*}{x})$ over the affine set

$$x_0 + \{p_0, p_1, \ldots, p_i\}$$

Proof

Let $\overset{*}{x} = x_0 + \sum_{j=0}^{i} t_j p_j$

In solving the linear system $Ax = b$ i.e. to find an approximate solution x^* for the system such that to minimize the error functional

$$E_\mu(\overset{*}{x}) = (x - \overset{*}{x},\ A^\mu(x - \overset{*}{x})) = (x - \overset{*}{x},\ A^{\mu-1}(Ax - A\overset{*}{x})) = (x - \overset{*}{x},\ A^{\mu-1}(Ax - b - (A\overset{*}{x} - b)))$$

Therefore by interpreting the above error functional, the error functional according to non-linear system $F(x) = 0$ is as follows

$$E_\mu(\overset{*}{x}) = (x - \overset{*}{x},\ J_{x^*}^{\mu-1}(F(x) - F(\overset{*}{x})))$$

1. for $\mu = 1$

$$E_\mu(\overset{*}{x}) = (x - \overset{*}{x},\ F(x) - F(\overset{*}{x})),\quad \overset{*}{x} = x_0 + \sum_{j=0}^{i} s_j p_j ,$$

$$E_\mu(\overset{*}{x}) = (x - x_0 - \sum_{j=0}^{i} s_j p_j,\ F(x) - F(x_0 - \sum_{j=0}^{i} s_j J_j p_j))$$

$$\simeq (x - x_0,\ F(x) - F(x_0)) - \sum_{j=0}^{i} \{2s_j(p_j, r_0) + s_j^2(p_j, J_j p_j)\}$$

$$\lesssim E_\mu(x_0) - \sum_{j=0}^{i} \{2s_j(p_j, r_j) + s_j^2(p_j, J_j p_j)\}$$

Therefore $E_\mu(x_i)$ is bounded below and it attains its minimum value when $s_j = (r_j, p_j)/(p_j, J_j p_j) = a_j$

Hence the minimum point is $\overset{*}{x} = x_0 + \sum_{j=0}^{i} a_j p_j = x_{i+1}$

2. for $\mu > 1$

$$E_\mu(\overset{*}{x}) = (x - \overset{*}{x},\ J_{x^*}^{\mu-1}(F(x) - F(\overset{*}{x}))) \simeq (x - x_0 - \sum_{j=0}^{i} s_j p_j, J_{x^*}^{\mu-1} (F(x) - F(x_0 + \sum_{j=0}^{i} s_j p_j)))$$

$$= (x - x_0 - \sum_{j=0}^{i} s_j p_j,\ J_{x^*}^{\mu-1}(F(x) - F(x_0) - \sum_{j=0}^{i} s_j J_j p_j)))$$

$$= (x - x_0,\ J_{x^*}^{\mu-1}(F(x) - F(x_0))) - \sum_{j=0}^{i} \{2s_j(p_j, J_{x^*}^{\mu-1} F(x_0)) + s_j^2(p_j, J_{x^*}^{\mu-1} J_j p_j)\}$$

$$= E_\mu(x_0) - \sum_{j=0}^{i} \{2s_j(p_j, J_{x^*}^{\mu-1} r_0) + s_j^2(p_j, J_{x^*}^{\mu-1} J_j p_j)\}$$

$$= E_\mu(x_0) - \sum_{j=0}^{i} \{2s_j(p_j, J_{x^*}^{\mu-1} r_j) + s_j^2(p_j, J_{x^*}^{\mu-1} J_j p_j)\}$$

Therefore for minimization we have

$$s_j = (p_j, J_{x^*}^{\mu-1} r_j)/(p_j, J_{x^*}^{\mu-1} J_j p_j) = a_j$$

Hence the minimum point is $\overset{*}{x} = x_0 + \sum_{j=0}^{i} a_j p_j = x_{i+1}$

Therefore, using the minimization theorem, we have that

$$x_i = x_0 + \sum_{j=0}^{i-1} s_j J_0^i r_0$$

$$= x_0 + P_{i-1}(J_0) r_0 \qquad 3.7.2$$

where $\{s_j\}_{j=0}$ are scalars and $P_{i-1}(J_0)$ is a polynomial of degree at most $i - 1$ in J_0.

Thus since the above methods defined in the preceding section minimize E_μ, over all iterative vectors, we have the following:

Since $x_i = x_0 + P_{i-1}(J_0) r_0$

$$x - x_i = x - x_0 - P_{i-1}(J_0) r_0$$

$$x_i = x_0 + \sum_{j=0}^{i-1} s_j J_0^j r_0 = x_0 + P_{i-1}(J_0) r_0$$

$$r_i = r_0 - P_{i-1}(J_0) r_0 = (I - P_{i-1}(J_0)) r_0$$

and $E_\mu(x_i) = (x - x_i, J_i^\mu(x - x_i)) = (x - x_i, \left(\frac{F(x) - F(x_i)}{(x - x_i)}\right)^\mu (x - x_i))$

$$= (r_i, J_i^{\mu-2} r_i)$$

$$= (R_i(A) r_0, J_i^{\mu-2} R_i(A) r_0)$$

$$R_i(A) = I - P_{i-1}(A)$$

where $R_i(A)$ belongs to the set of polynomials satisfying $R_m(0) = 1$. Therefore these methods choose the particular polynomial $R_i \in R_i'$, that minimizes the functional, i.e.

$$E_\mu(x_i) = \min_{R_i \in R_i'} (R_i(J_i)r_0, J_i^{\mu-2}R_i(J_i)r_0)$$

Since A is symmetric, it has N-orthonormal eigenvectors $\{v_j | 1 \leq j \leq N\}$ where $Av_j = \lambda_j v_j \qquad j = 1,2,\ldots,N$ and where $\{\lambda_j\}_{j=1}^N$ are the eigenvalues of A

Since $r_0 = \sum_{j=1}^N t_j v_j$ for some scalars $\{t_j\}_{j=1}^N$

we have that $R_i(A)r_0 = \sum_{j=1}^N t_j R_i(A)v_j = \sum_{j=1}^N t_j R_i(\lambda_j)v_j$

Therefore

$$\begin{aligned}
E_\mu(x_i) &= \min_{R_i \in R_i'} \left(\sum_{j=1}^N t_j R_i(\lambda_j)v_j, J_i^{\mu-2} \sum_{j=1}^N t_j R_i(\lambda_j)v_j\right) \\
&= \min_{R_i \in R_i'} \left(\sum_{j=1}^N t_j R_i(\lambda_j)v_j, \sum_{j=1}^N t_j R_i(\lambda_j)\lambda_j^{\mu-2}v_j\right) \\
&= \min_{R_i \in R_i'} \left(\sum_{j=1}^N t_j^2 R_i^2(\lambda_j)\lambda_j^{\mu-2}\right) \\
&\leq \min_{R_i \in R_i'} \left(\max_{1\leq j\leq N} |R_i(\lambda_j)|\right)^2 \left(\sum_{j=1}^N t_j^2 \lambda_j^{\mu-2}\right) \\
&\leq \min_{R_i \in R_i'} \left(\max_{1\leq j\leq N} |R_i(\lambda_j)|\right)^2 \left(\sum_{j=1}^N t_j v_j, \sum_{j=1}^N t_j \lambda_j^{\mu-2} v_j\right) \\
&= \min_{R_i \in R_i'} \left(\max_{1\leq j\leq N} |R_i(\lambda_j)|\right)^2 \left(\sum_{j=1}^N t_j v_j, J^{\mu-2} \sum_{j=1}^N t_j v_j\right) \\
&= \min_{R_i \in R_i'} \left(\max_{1\leq j\leq N} |R_i(\lambda_j)|\right)^2 (r_0, J^{\mu-2} r_0) \\
&= Q_i^2 E_\mu(x_0)
\end{aligned}$$

where $Q_i = \min_{R_i \in \acute{R}_i} \max_{1 \leq j \leq N} |R_i(\lambda_j)|$

Theorem 3.7.3

At the i-th step of the iterations, the iterative variational method satisfies

$$||x - x_i||_{J_i^\mu} \leq Q_i ||x - x_0||_{J_i^\mu}$$

Corollary 3.7.2

If r_0 has a non-zero component only for $k \leq N$ eigenvectors of J_i corresponding to $m \leq k$ distinct eigenvalues, then the methods converge in at most m-iterations to the unique solution of $F(x) = 0$.

Proof

Assume that r_0 has non-zero components only along the eigenvectors $v_1, v_2, \ldots, v_k$ of J_i, then

$$r_0 = \sum_{j=1}^{k} t_j v_j \quad \text{for some scalars } \{t_j\}_{j=1}^{k}$$

$||x - x_i||_{J_i^\mu} \leq Q_i ||x - x_0||_{J_i^\mu}$, where $Q_i = \min_{R_i \in \acute{R}_i} \max_{1 \leq j \leq N} R_i(\lambda_j)$

Assume that among the eigenvalues $\lambda_1, \ldots, \lambda_k$ only $\lambda_1, \lambda_2, \ldots, \lambda_m$ are distinct, then

$$Q_i = \min_{R_i \in \acute{R}_i} \max_{1 \leq j \leq N} |R_i(\lambda_j)|$$

The polynomial $R_m(y) = \prod_{j=1}^{m} \left(\frac{y - \lambda_j}{\lambda_j}\right)$ belongs to $\acute{R}_m$. Moreover it vanishes at $\{\lambda_j\}_{j=1}^{m}$ thus $Q_m = 0$ which implies that

$$||x - x_i||_{J_i^\mu} = 0 \quad \text{for some } i \leq m$$

3.8 Non linear conjugate gradient method

In the last section we studied a class of variational methods for solving non-linear problems with symmetric jacobians. Here we are going to consider the solution of non-linear operator with symmetric and positive definite jacobians.

In the last section we defined $R_i(\lambda)$ as the polynomial of degree i of least deviation from zero on the eigenvalue spectrum of J_i normalized so that $R_i(0) = 1$.

For general eigenvalue distributions we cannot find a minimum polynomial and cannot evaluate Q_i, but the upper bounds on Q_i can be obtained.

Let $[a,b]$, $a,b > 0$ be an inverval known to contain all the eigenvalues of J_i, choose $R_i(\lambda)$ to be the unique polynomial in $R_i'(\lambda)$ that minimizes the deviation from zero on $[a,b]$, then the normalized Tchebyshev polynomial on $[a,b]$, i.e.

$$R_i(\lambda) = \frac{T_i\left(\frac{-2\lambda + a + b}{b - a}\right)}{T_i\left(\frac{b + a}{b - a}\right)} \qquad 3.8.1$$

where $T_i(z) = \cos(i \arccos z)$ is the i-th Tchybshev polymonial in z, therefore

$$Q_i \leq 2\left(\frac{1 - \sqrt{\alpha}}{1 + \sqrt{\alpha}}\right)^i \quad \text{for } i \geq 0$$

where $\alpha = a/b$, $k(J_i) = b/a$

Theorem 3.8.1

At i-th step of iteration, $i \geq 0$

$$||x - x_i||_{J_i^\mu} \leq 2\left(\frac{1 - \sqrt{\alpha}}{1 + \sqrt{\alpha}}\right)^i ||x - x_0||_{J_i^\mu}, \text{ where } \alpha = \frac{1}{k(J_i^\mu)}$$

By following the same policy as above we can interpret the results as has been proposed by Axelsson (A6) to obtain bound on Q_i by assuming that the eigenvalues of J_i (of A in Axelsson, to solve Ax = b) are distributed over two intervals on the positive real axis.

One of the methods which can be generated from the variational methods is the "conjugate gradient method" to solve the non-linear operator F(x) = 0, which is one of the variational scheme when $\mu = 1$ and here we present the original algorithm presented by J.W. Daniel (D2), which is the version of variational method when $\mu = 1$.

Algorithm 3.8.1

Non-linear conjugate gradient to solve F(x) = 0

1. Choose x_0

   $r_0 = -F(x_0)$

   $p_0 = r_0$

2. $a_i = (p_i, r_i)/(p_i, J_i p_i)$

   $x_{i+1} = x_i + a_i p_i$

   $r_{i+1} = - F(x_{i+1})$

   $p_{i+1} = r_{i+1} + b_i p_i$

   $b_i = - (r_{i+1}, J_{i+1} p_i)/(p_i, J_{i+1} p_i)$

3. if convergence achieved (STOP)

   otherwise set i = i + 1 and go to 2

   (See Note on p.69)

As has been proved before the rate of convergence of the iterative variational method is a monotone decreasing function of $k(J_i)$ (in linear case k(A)). Hence we want to improve the rate by decreasing the condition number of the iteration matrix s.t. $k(A') < k(A)$ where A is a certain iterative matrix.

As has been described there are several ways of reducing the condition number of the iterative matrix (in solving the non-linear system the corresponding iterative matrix is the Jacobian of the system) one can use all the mentioned splitting of the matrix which has been described before to reduce the condition number.

Algorithm 3.8.2

The Preconditioned Variational Method

1. Choose x_0

   Compute $r_0 = F(X_0)$

   Solve $M_0 z_0 = r_0$

   $p_0 = z_0$

2. Iteration i

   $a_i = (r_i, (M_i^{-1} J_i)^{\mu-1} z_i) / (p_i, J_i (M_i^{-1} J_i)^{\mu-1} p_i)$

   $x_{i+1} = x_i + a_i p_i$

   $r_{i+1} = F(x_{i+1}) \simeq r_i + a_i J_i p_i$

   solve $M_i z_{i+1} = r_{i+1}$

   $b_i = (r_{i+1}, (M_i^{-1} J_i)^{\mu-1} z_{i+1}) / (r_i, (M_i^{-1} J_i)^{\mu-1} z_i)$

   $p_{i+1} = z_{i+1} + b_i p_i$

3. if convergence achieved (STOP)

   otherwise set i = i + 1 and go to 2

Similarly to theorem 3.6.1 we have the following theorem for the error bound

Theorem 3.8.2

At the i-th iterative step, we have the following

$$||x - x_i||_{J_i(M_i^{-1}J_i)^{\mu-1}} \leq 2\left(\frac{1 - \sqrt{\alpha}}{1 + \sqrt{\alpha}}\right)^i ||x - x_0||_{J_i(M_i^{-1}J_i)^{\mu-1}}$$

The preconditioned conjugate gradient method takes its simplest form by taking $\mu = 1$, which will produce the algorithm presented by P. Concus, H. Goluband, P. O'Leary (C9), the algorithm is as follows

Algorithm 3.8.3

Non-linear Preconditioned Conjugate Gradient

1. Choose x_0

   Compute $r_0 = F(x_0)$

   Solve $M_0 z_0 = r_0$

   $p_0 = r_0$

2. $a_i = (r_i, z_i)/(p_i, J_i p_i)$

   $x_{i+1} = x_i + a_i p_i$

   $r_{i+1} = F(x_{i+1}) \simeq r_i + a_i J_i p_i$

   solve $M_{i+1} z_{i+1} = r_{i+1}$

   $b_i = (r_{i+1}, z_{i+1})/(r_i, z_i)$

   $p_{i+1} = z_{i+1} + b_{i+1} p_i$

3. if x_{i+1} is of desired accuracy, STOP

   otherwise set $i = i + 1$ and go to 2

   (See Note below)

P. Concus et al (C9) used different kind of preconditioned.

In the application of algorithm 3.8.3 to solve the system of linear equations generated from the discretization of mildly non-linear elliptic equations, we used the modified Stone factorization (B10).

Note: The evaluation of the Jacobian is time consuming if it is not simply diagonal. The evaluation can be avoided by differences of successive residuals.

The starting vectors were derived by generating various numbers for each component of the initial vectors, each in the range [0.05, M] where M is the maximum boundary value of the problem. The same starting vector was used for different methods applied to any given problem.

CHAPTER 4 Numerical Results and Discussions

4.1 Introduction

Complexity is defined as a measure of cost, bearing on costs depending on the model under analysis.

In this Chapter we present results of numerical experiments comparing the performance of the following algorithms.

1. S.I.P. (Linearized technique)
2. Non-Linearized S.I.P.
3. Linear Conjugate Gradient
4. Non-Linear Conjugate Gradient
5. Preconditioned non-linear conjugate gradient

On the following problems (see H8)

problem 1: $u_{xx} + u_{yy} = \text{Exp}(u)$, $u \in R$, B.C.: $u = x + 2y$, $(x,y) \in \partial R$

problem 2: $u_{xx} + u_{yy} = u^3$, $u \in R$, B.C.: $u = x + 2y$, $(x,y) \in \partial R$

problem 3: $u_{xx} + u_{yy} = u^3$, $u \in R$, B.C.: $u = 0$, $(x,y) \in \partial R$

The domain R is:

$$R = \{(x,y) \mid 0 \leq x \leq 1,\ 0 \leq y \leq 1\}$$

With regard to the following cost factors

1. Number of arithmetics
2. Size of storage required
3. Number of total iterations required for acceptable convergence
4. Time required (Time in seconds, using ICL 2980, operating system VME/B, version 5 x 32 and main store 4 Mbytes)

4.2 Number of arithmetics

Number of arithmetics is defined, the number of multiplications required to perform one iterative step. To make a decision on the efficiency of a certain algorithm is to consider the total number of arithmetics required to solve the model problem.

To be precise, we suggest the following classification

1. Information Arithmetics: In general this includes all the arithmetics required to generate the system of non-linear equations. But in some techniques, for instance S.I.P., we require also a matrix factorization LU to be calculated.

2. Linearization Arithmetics: Solving a system of non-linear equations by any linearization technique requires an evaluation of the non-linear operator.

An extra work might be essential in some techniques, for instance in linear conjugate gradient the current residual is required at the first step of the algorithm.

3. Algorithm Arithmetics: This is the most important factor - how many arithmetics are required in just one step of the algorithm?

In table 4.1 we list all the arithmetics required by different algorithms.

4.3 Size of storage

This is also an important factor in making a decision on the efficiency of any algorithm.

The total size of storage can be considered with regard to the various stages of operating any algorithm as:

Algorithm / Arithmetics	S.I.P. (Linearization)	S.I.P. (Non-Linearization	Linear Conjugate Gradient	Non-Linear Conjugate Gradient	Preconditioned Non-Linear Conjugate Gradient
Information Arithmetics	1. Matrix Factorization 2. Product of LU (LU = A + B)	1. Matrix Factorization 2. Product of LU	-	-	-
Linearization Arithmetics	Linearization	-	1. Linearization 2. $5N^2 - 2N$	-	-
Algorithm Arithmetics	$12N^2 - 6N - 2$ + Backward and Forward solver	$12N^2 - 6N - 2$ + Backward and Forward solver	$(5N^2 - 2N) + 6N^2$ or $(5N^2 - 2N) + 7N^2$	1. $5N^2 - 2N$ + Evaluation of non-linear operator 2. $(5N^2 - 2N) + 3N^2$ 3. $(5N^2 - 2N)$ + Evaluation of non-linear operator 4. $(5N^2 - 2N) + 2N^2$ 5. Evaluation of the Jacobian 6. $(5N^2 - 2N) + 2N^2$	1. $(5N^2 - 2N)$ + Evaluation of the non-linear operator 2. Evaluation of the Jacobian 3. Jacobian Factorization 4. $(5N^2 - 2N) + 6N^2$

Table 4.1 The arithmetics required by methods 1-5.

1. The information storage: This includes the store units required for the information vectors which includes

   1-a The system of non-linear equations generated from the application of the finite difference approximations.

   1-b In some methods for instance S.I.P. and preconditioned non-linear conjugate gradient is required an approximate factorization for the linearized system or the approximate factorization of the Jacobian of the non-linear system and some extra storage for the residual or the directional vector is required as well.

2. The algorithm store: which includes all the storage by the application of a certain algorithm.

3. The convergence store: Any extra storage might be required for convergence checking.

In Table 4.2 we list all the storage required by each algorithm.

4.4 Number of iterations and time

Problems 1, 2 and 3 have been solved by non-linearized S.I.P. and various semblances of linearized S.I.P., figures 4.1-9 show the variation in number of iterations and $\alpha(\alpha \in [0,1])$.

Tables 4.3-9 show the time and number of iterations required to solve problems 1, 2 and 3 by using the following convergence checking

1. $||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$
2. $||x_{i+1} - x_i||_\infty / ||x_{i+1}||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$

Algorithm / Store	S.I.P. (Linearization)	S.I.P. (Non-linearization)	Conjugate gradient (for linear system)	Non-linear conjugate gradient	Preconditioned non-linear conjugate gradient (Stone factorisation)
Information store	14	14	6	7	10
Inner itreation information store	3	-	5	-	-
Algorithm store	7	9	6	13	13
Convergence store	1	1	1(0)	0	0
Outer iteration convergence store	-	-	1	-	-

Table 4.2 A complete table of the store units required by each algorithm which has been used

Solving the system of non-linear equations by non-linearized S.I.P. (with $||x_{i+1} - x_i||_\infty < \varepsilon$, for convergence checking) requires less iterations and less time than any semblance of linearized S.I.P.

Also $||x_{i+1} - x_i||_\infty / ||x_{i+1}||_\infty < \varepsilon$ is more useful convergence criteria than $||x_{i+1} - x_i||_\infty < \varepsilon$, provided $||x_{i+1}||_\infty$ is not too close to zero, because it consumes less iterations and time than using the convergence criteria $||x_{i+1} - x_i||_\infty < \varepsilon$.

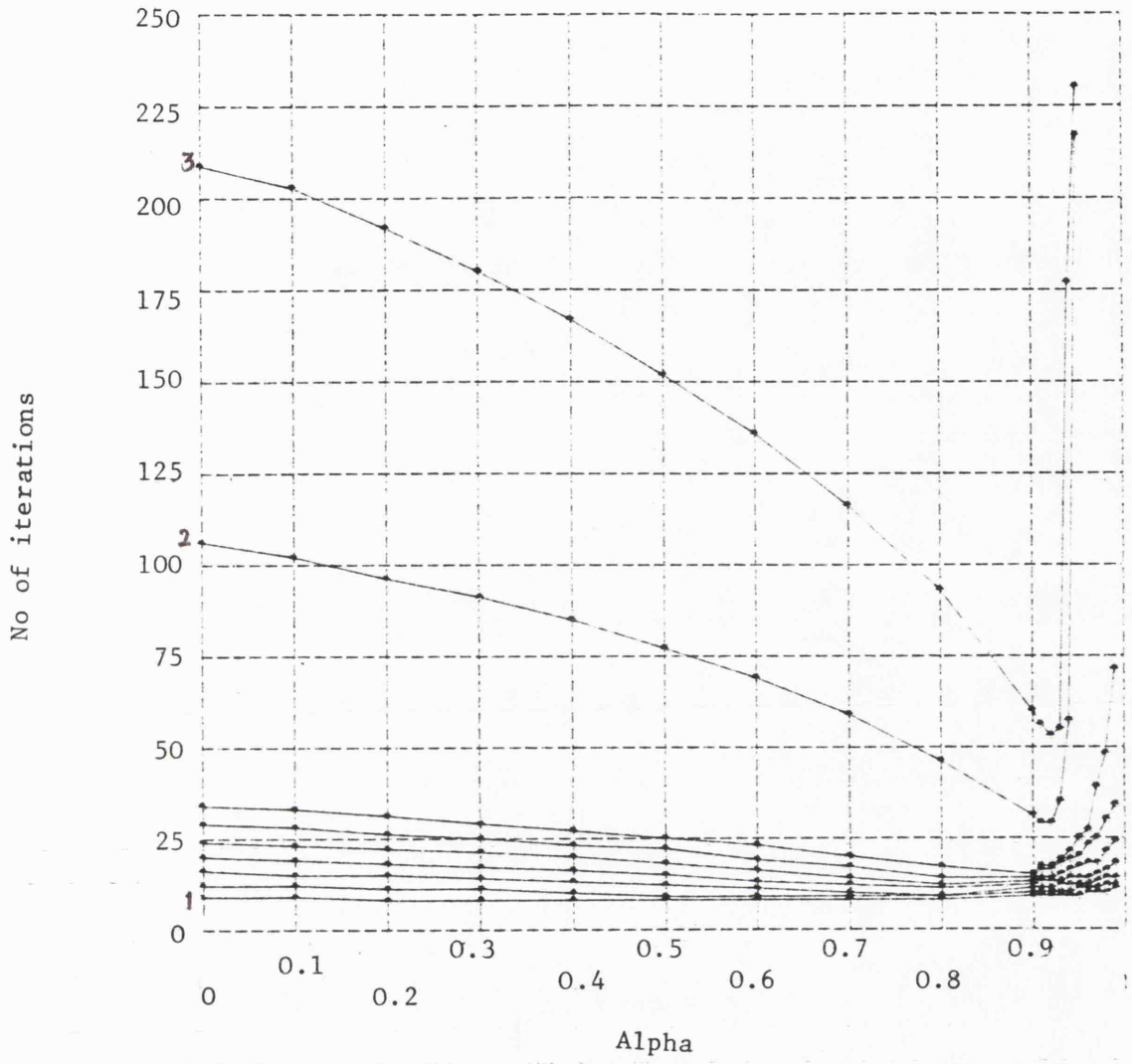


Fig. 4.1

The variation in the number of iterations with respect to $\alpha \in [0,1]$ to solve problem 1 by non-linearized S.I.P.

1. For a system of dimension = 16
2. For a system of dimension = 900
3. For a system of dimension = 1600

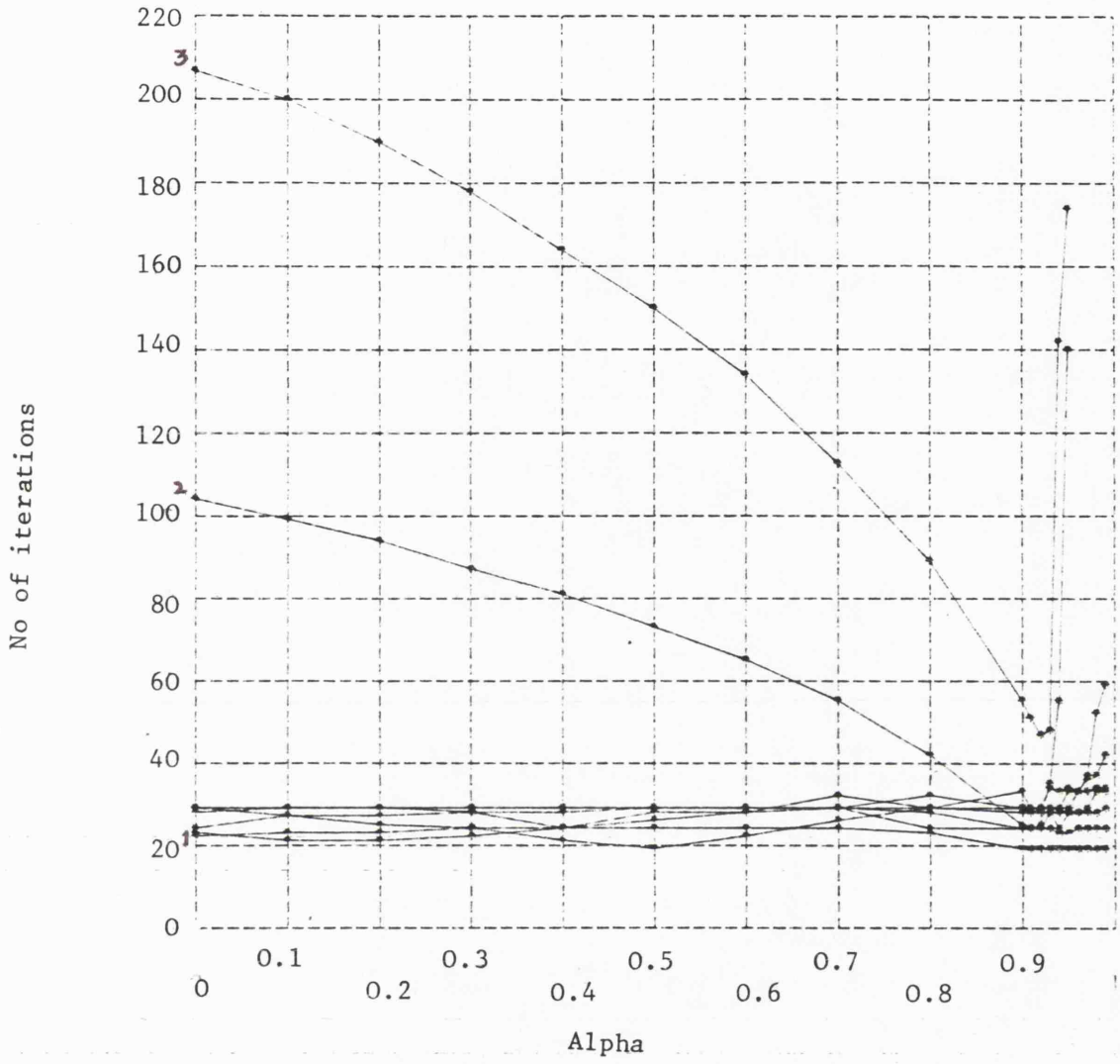


Fig. 4.2

The variation in the number of iterations with respect to $\alpha \in [0,1]$ to solve problem 1 by linearized S.I.P. every 5 iterations

1. For a system of dimension = 16
2. For a system of dimension = 900
3. For a system of dimension = 1600

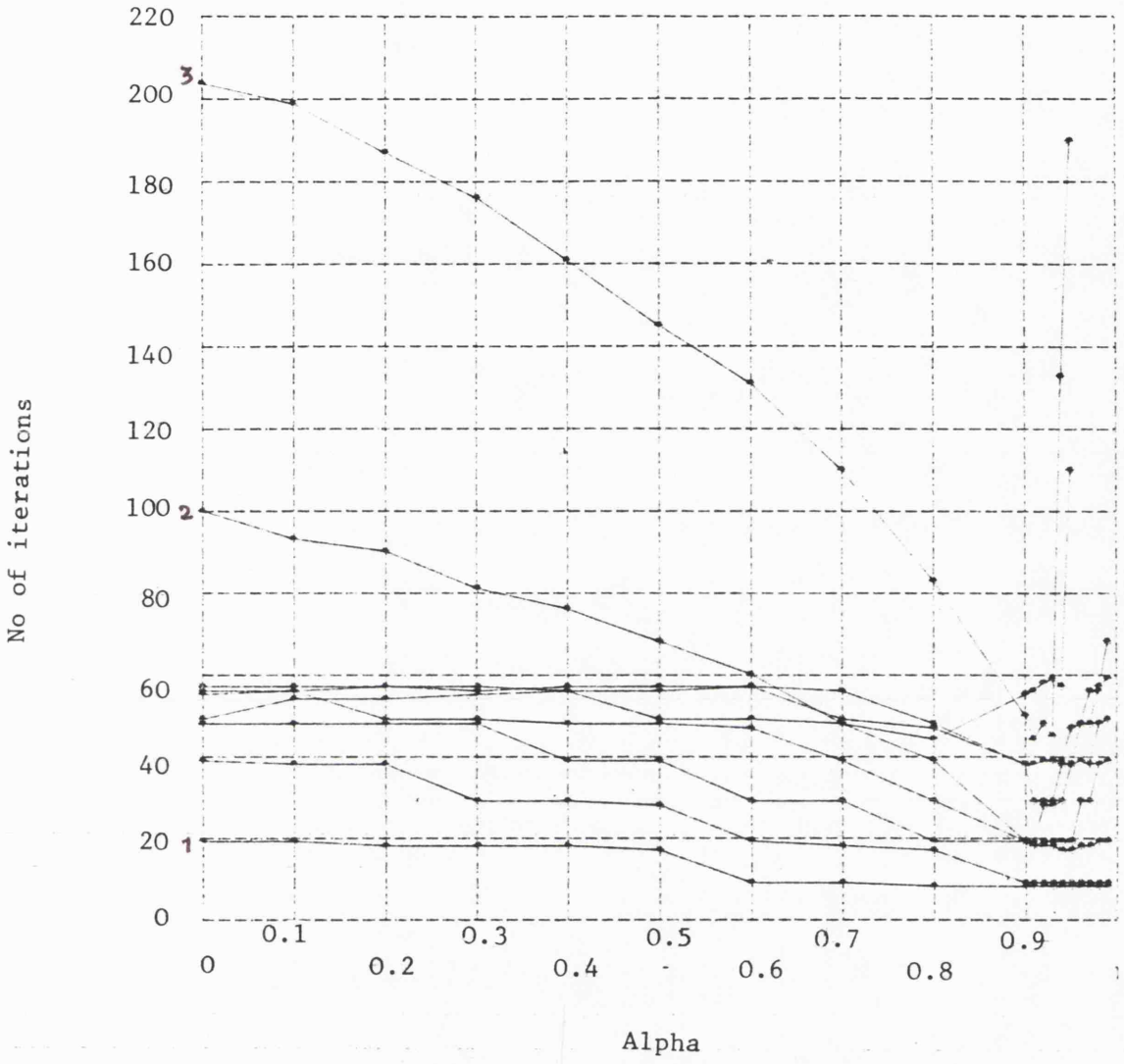


Fig. 4.3

The variation in the number of iterations to solve problem 1 with respect to $\alpha \in [0,1]$ by linearized S.I.P. every 10 iterations

1. For a system of dimension = 16
2. For a system of dimension = 900
3. For a system of dimension = 1600

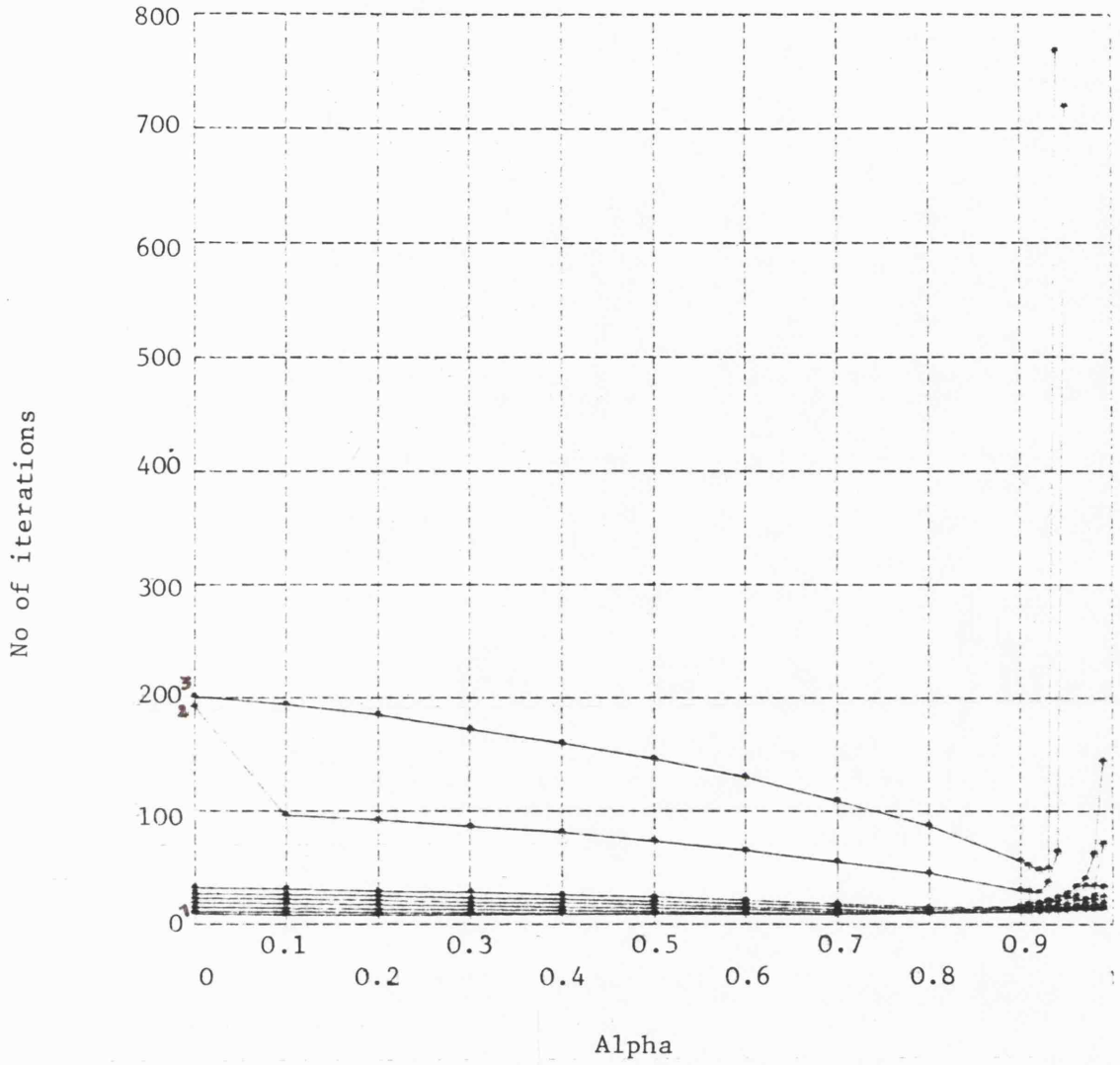


Fig. 4.4

The variations in the number of iteration to solve problem 2 with respect to $\alpha \in [0,1]$ by non-linearized S.I.P.

1. For a system of dimension = 16
2. For a system of dimension = 900
3. For a system of dimension = 1600

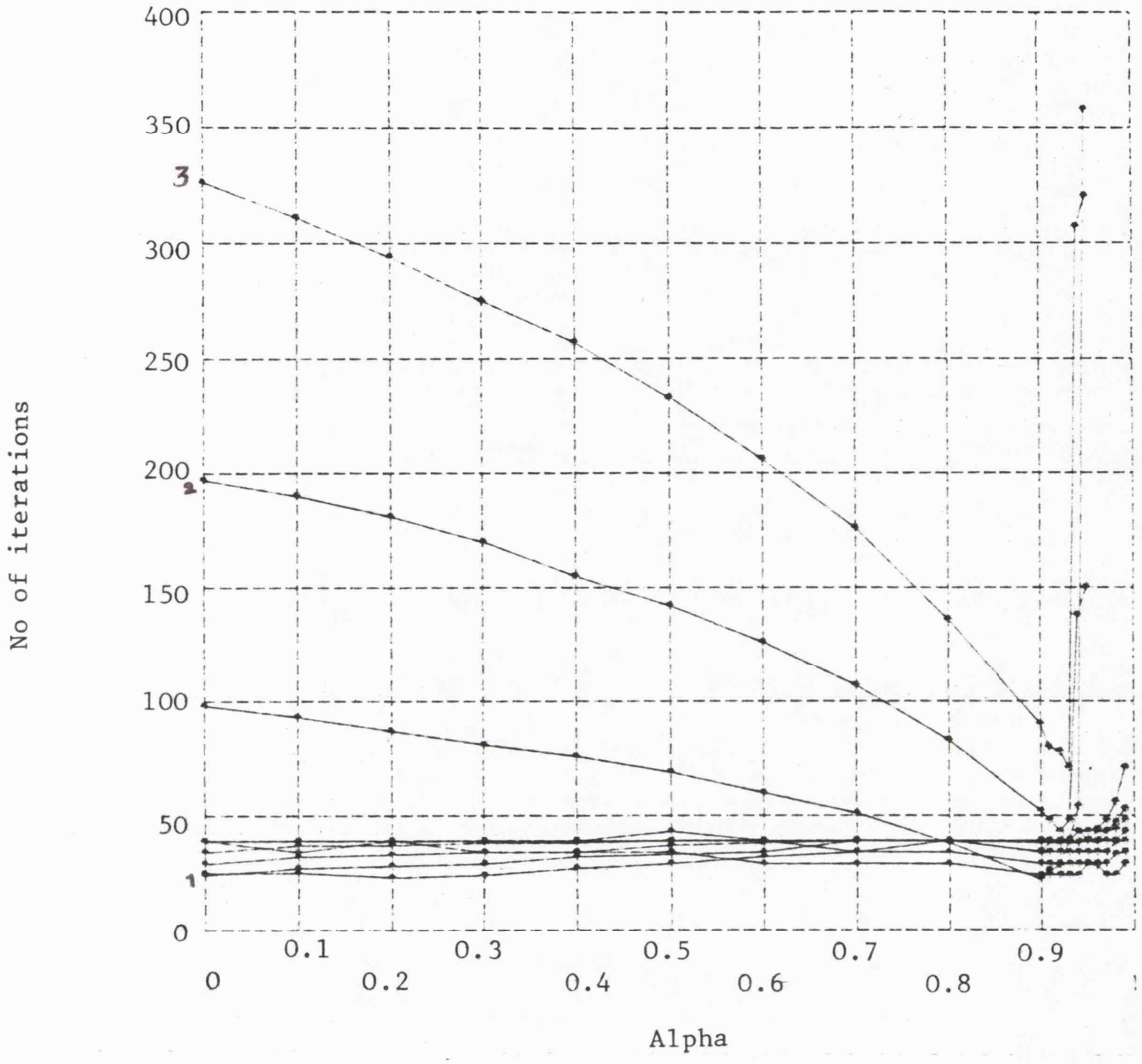


Fig. 4.5

The variations in the number of iterations to solve problem 2 with respect to $\alpha \in [0,1]$ by linearized S.I.P. every 5 iterations

1. For a system of dimension = 16
2. For a system of dimension = 900
3. For a system of dimension = 1600

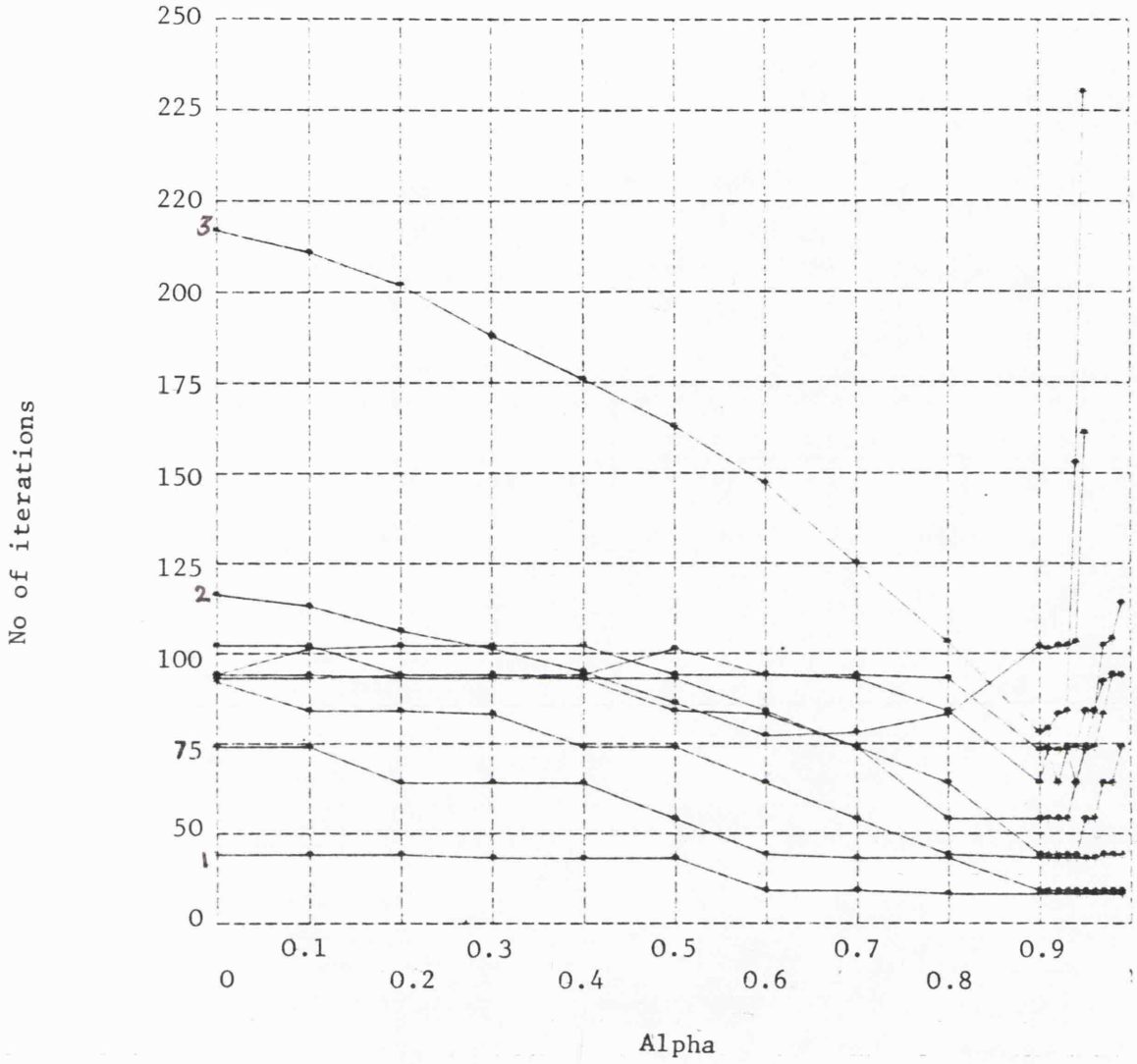


Fig. 4.6

The variations in the number of iterations to solve problem 2 with respect to $\alpha(\alpha \in [0,1])$, by linearized S.I.P. every 10 iterations

1. For a system of dimension = 16
2. For a system of dimension = 900
3. For a system of dimension = 1600

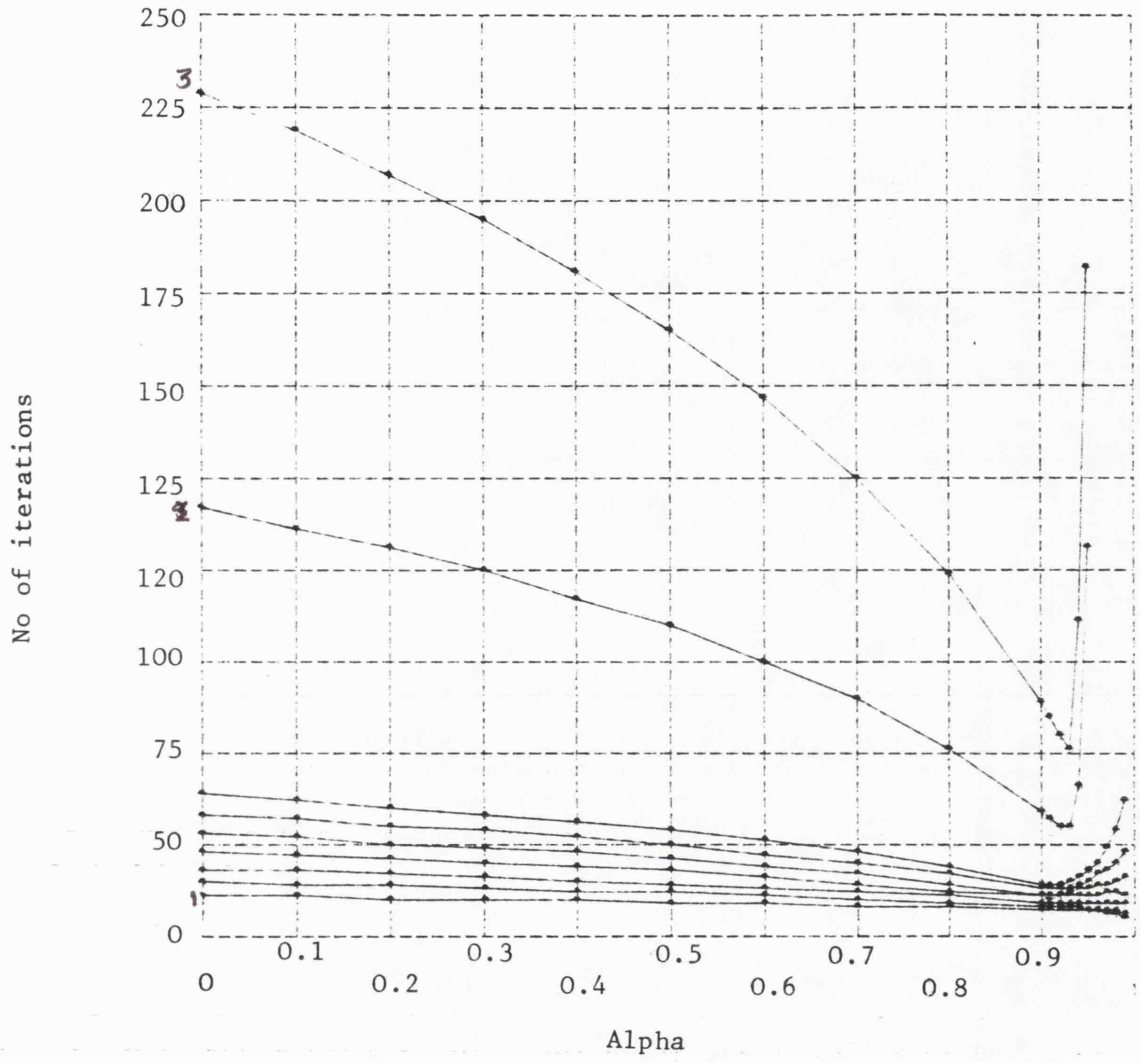


Fig. 4.7

The variation in the number of iterations to solve problem 3 with respect to $\alpha(\alpha \in [0,1])$ by non-linearized S.I.P.

1. For a system of dimension = 16
2. For a system of dimension = 900
3. For a system of dimension = 1600

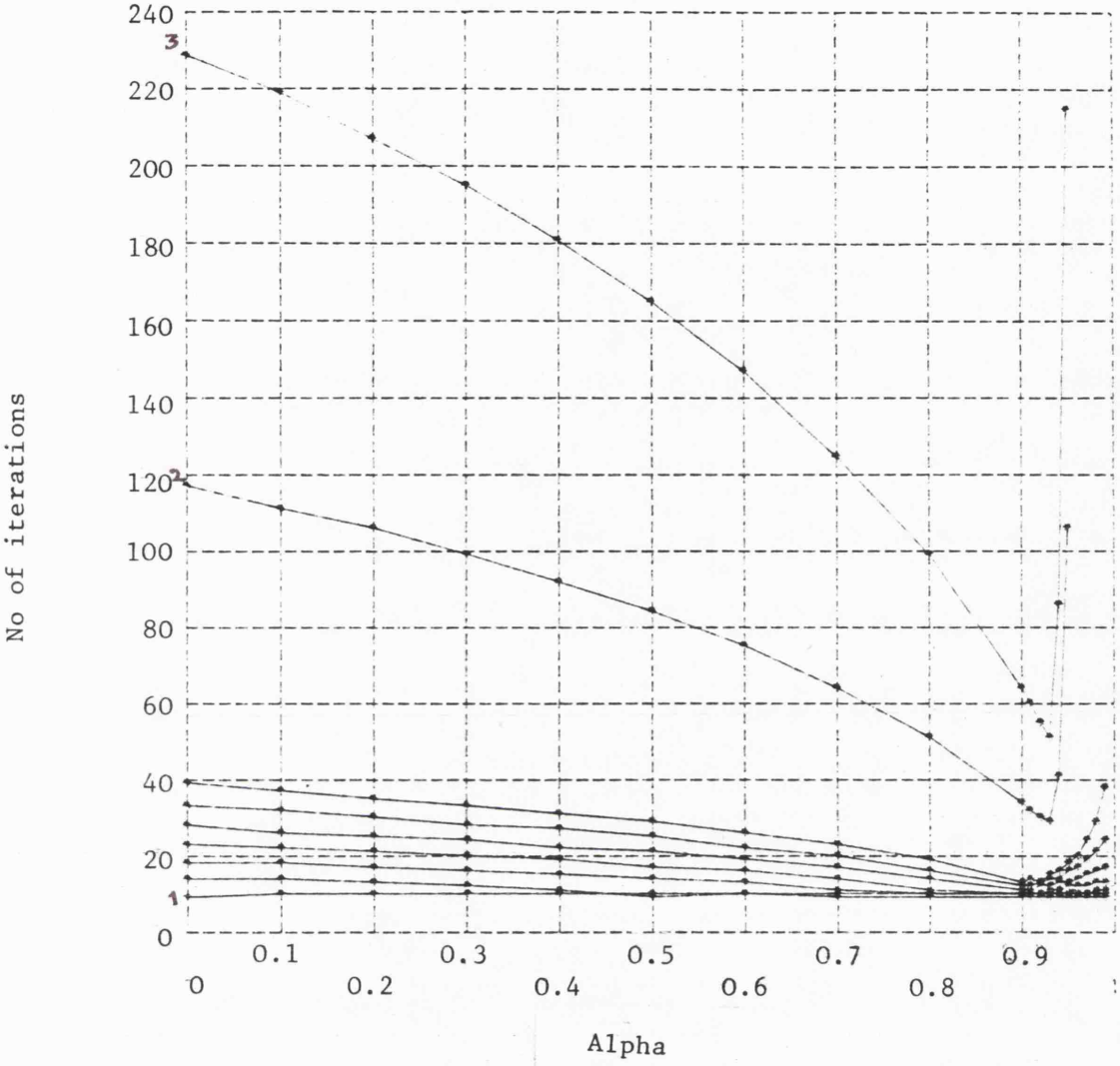


Fig. 4.8

The variation in the number of iterations required to solve problem 3 with respect to $\alpha(\alpha \in [0,1])$ by linearized S.I.P. every 5 iterations

1. For a system of dimension = 16

2. For a system of dimension = 900

3. For a system of dimension = 1600

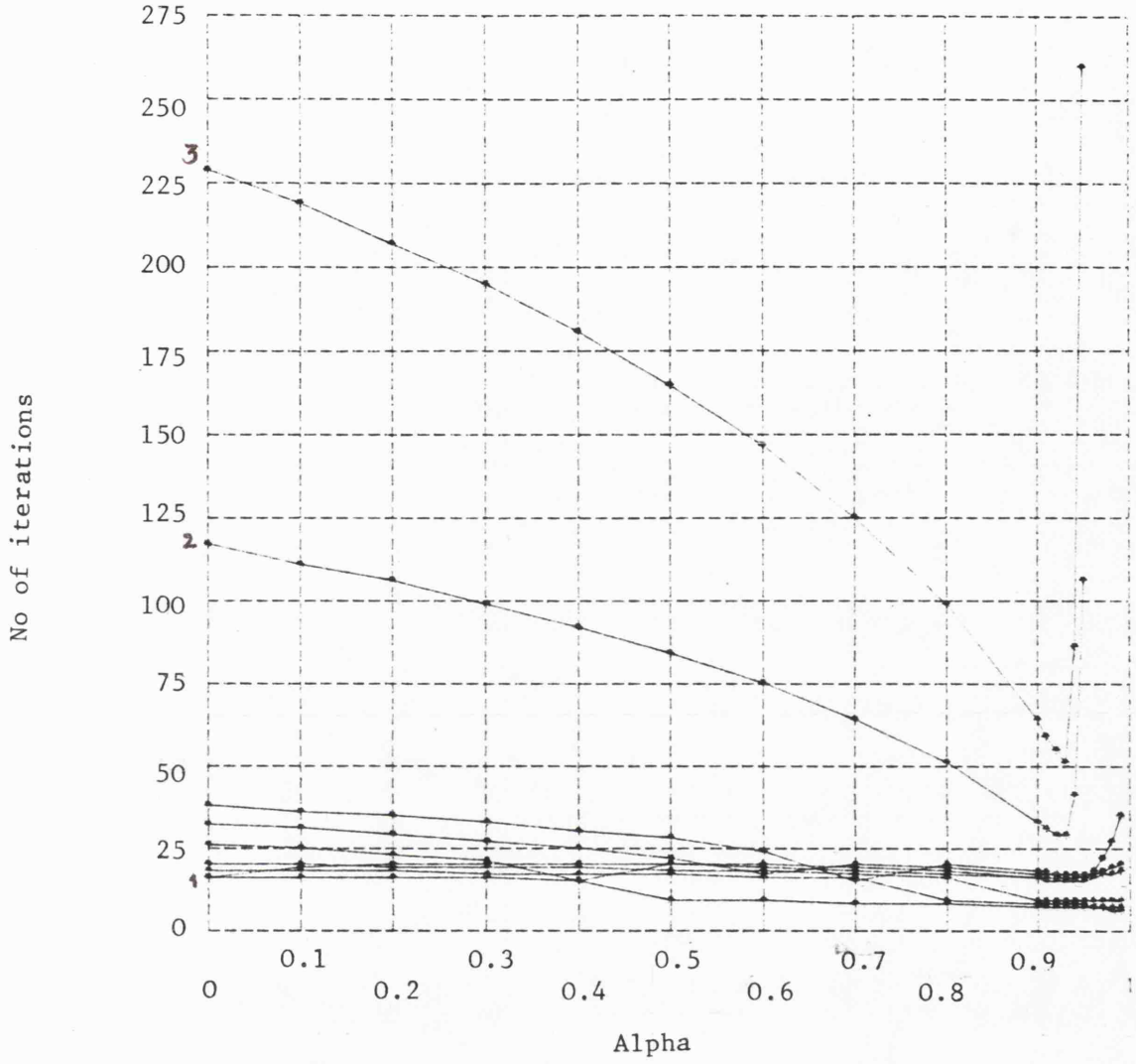


Fig. 4.9

The variations in the number of iterations to solve problem 3 with respect to $\alpha(\alpha \in [0,1])$ by linearized S.I.P. every 10 iterations

1. For a system of dimension = 16
2. For a system of dimension = 900
3. For a system of dimension = 1600

Index of tables 4.3-4.8

A Linearized S.I.P. and to restart the outer iteration at the time the convergence of the linearized system is achieved

B Linearized S.I.P. and to restart the outer iteration every 5 iterations

C Linearized S.I.P. and to restart the outer iterations every 10 iterations

D Non linearized S.I.P.

Problem Dimension	A	B	C	D
81	63	21	29	14
100	68	21	38	15
400	132	24	38	29
900	190	47	44	53

Table 4.3 Number of iterations required to solve problem 1 by methods A, B, C and D (with $||x_{i+1} - x_i||_\infty < \varepsilon, \varepsilon = 10^{-5}$, convergence criteria)

Problem Dimension	A	B	C	D
81	84	24	39	13
100	58	23	48	15
400	176	22	52	28
900	311	43	53	48

Table 4.4 Number of iterations required to solve problem 2 by methods A, B, C and D (with the convergence criteria $||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$)

Problem Dimension	A	B	C	D
81	6	12	17	13
100	7	13	16	14
400	12	29	29	29
900	18	51	51	51

Table 4.5 Number of iterations required to solve problem 3 by methods A, B, C and D (with the convergence criterion $||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$)

Problem Dimension	Problem 1		Problem 2	
	C	D	C	D
81	19	13	29	12
100	29	12	38	14
400	37	25	40	25
900	36	43	39	42

Table 4.6 Number of iterations to solve problems 1 and 2 by methods C and D (with the convergence criteria $||x_{i+1} - x_i||_\infty / ||x_{i+1}||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$)

Problem dimension	A	B	C	D
81	0.35411	0.11121	0.13022	0.10242
100	0.49575	0.13667	0.20793	0.13358
400	2.82292	0.59688	0.93538	0.98827
900	8.53240	2.60457	2.31439	3.98541

Table 4.7 Time required to solve problem 1 by methods A, B, C and D (with the convergence criteria $||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$)

Problem dimension	A	B	C	D
81	0.45276	0.18458	0.16728	0.18458
100	0.36199	0.22540	0.25126	0.22540
400	3.6656	0.50705	1.04246	0.50705
900	14.09351	2.19119	2.63280	2.19119

Table 4.8 Time required to solve problem 2 by methods A, B, C and D (with convergence criteria $||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$)

Problem Dimension	Problem 1		Problem 2	
	S.I.P. (non-linearization)	S.I.P. (linearization and restart every 10 iterations	S.I.P. (non-linearization)	S.I.P. (linearization and restart every 10 iterations
81	0.09487	0.09518	0.06493	0.13609
100	0.11898	0.17537	0.09205	0.21706
400	0.85274	0.86243	0.61987	0.84073
900	3.350585	1.90170	2.30357	1.95442

Table 4.9 The time required to solve problems 1 and 2 by methods C and D (with convergence criteria $||x_{i+1} - x_i||_\infty / ||x_{i+1}||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$)

From the performance of linear conjugate gradient method and non linear conjugate gradient method (with various specified fixed number of iterations for Jacobian evaluation) to solve problems 1, 2 and 3 with the convergence criteria $||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$ (see tables 4.10-13), using non linear conjugate gradient with evaluating the Jacobian every 5-iterations require less iterations and time than the other guises of non linear conjugate gradient method. Specifically in a non linear conjugate gradient with evaluating the Jacobian every 10-iterations, requires more iterations and time, produces directional vectors $\{p_i\}$ some of which deviate "significantly" from the correct direction.

The same behaviour occurred in non linear conjugate gradient technique with the convergence criteria $||R_i||_2 < \varepsilon$ (R_i as the current residual, $\varepsilon = 10^{-5}$).

Also, in linear conjugate gradient method, $||R_i||_2 < \varepsilon$ is a more useful convergence criteria than $||x_{i+1} - x_i||_\infty < \varepsilon$ (see tables 4.14-17) because it consumes less iterations and time than non linear conjugate gradient method (and other relevant forms of non linear conjugate gradient method.

Problems 1, 2 and 3 have been solved by preconditioned non linear conjugate gradient (see tables 4.18, 19 and 20) with factorizing the Jacobian at every iteration, 5-iterations and every 10-iterations.

Refactorizing the evaluated Jacobian at these different levels did not reduce the number of iterations but it reduced the total time required, for instance, refactorizing the Jacobian at every 10-iterations consumes less time than refactorizing at every iteration and 5-iterations.

Comparatively with the other techniques, preconditioned non linear conjugate gradient method requires less iterations than linear (non linear) conjugate gradient method and linearized (non linearized) S.I.P.

But the time consumed by nonlinearized S.I.P. is less than the time consumed by preconditioned non linear conjugate gradient method, which consumes less time than linear (non linear) conjugate gradient method.

Eventually, considering the storage and arithmetics required, the total work required by preconditioned non linear conjugate gradient method is more than the non linearized S.I.P.

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	162	2.53792	155	1.60371	177	1.82662
100	191	3.64691	183	2.29325	206	2.54173
400	616	45.52620	585	27.30873	657	30.28906
900			1127	116.10680	1295	133.236

Table 4.10 No of iterations and time required to solve problems 1, 2 & 3 by conjugate gradient method and re-evaluating the Jacobian every iteration ($||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$, for convergence checking)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	104	1.07853	79	0.62826	67	0.52895
100	99	1.24376	94	0.90234	72	0.68358
400	239	11.65172	243	8.84333	222	7.97190
900	414	45.03780	349	28.2379	317	25.36894
1600	615	118.61226	795	113.9635	599	84.99241

Table 4.11 Number of iterations and time required to solve problems 1, 2 and 3 by non linear conjugate gradient method and re-evaluate the Jacobian every 5-iterations ($||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$ for convergence checking)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	112	1.10879	87	0.66506	92	0.71029
100	132	1.58677	84	0.77391	96	0.89161
400	260	11.90041	336	11.69024	259	9.03954
900	470	48.04312	518	40.31071	499	38.7283

Table 4.12 Number of iterations and time required to solve problems 1, 2 and 3 by non linear conjugate gradient method and re-evaluating the Jacobian every 10 iterations ($||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$ for convergence checking)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	97	0.53192	126	0.66950	23	0.13022
100	106	0.69166	140	0.88619	24	0.16264
400	178	4.11318	234	5.25426	47	1.10280
900	257	12.96755	339	16.74306	69	3.52727
1600	336	29.65411	439	38.20345	93	8.32014

Table 4.13 Number of iterations and time required to solve problems 1, 2 and 3 by linear conjugate gradient method ($||x_{i+1} - x_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$, for convergence checking)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	129	2.05986	123	1.23953	139	1.42059
100	149	2.89645	143	1.73711	155	1.89347
400	407	30.16924	393	17.94501	419	19.24257
900	699	115.59140	681	69.5335	697	71.445017

Table 4.14 Number of iterations and time required to solve problems 1, 2 and 3 by non linear conjugate gradient method and evaluate the Jacobian every iteration

($||R_i||_2 < \varepsilon$, $\varepsilon = 10^{-5}$, for convergence checking)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	95	1.01112	68	0.53624	61	0.48955
100	88	1.13730	81	0.76939	71	0.68221
400	216	10.72319	193	6.94329	221	7.99438
900	309	34.151814	264	21.13054	266	21.41138
1600	464	91.62886	536	107.3599	391	56.03732

Table 4.15 Number of iterations and time required to solve problems 1, 2 and 3 by non linear conjugate gradient method and re-evaluate the Jacobian every 5-iterations

($||R_i||_2 < \varepsilon$, $\varepsilon = 10^{-5}$, for convergence checking)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	111	1.12533	98	0.74979	81	0.62153
100	129	1.58266	77	0.71034	111	1.02476
400	241	11.30343	321	11.23829	303	10.61646
900	411	42.94211	463	36.21605	383	29.84511
1600	627	116.740	543	75.63634	533	73.72124

Table 4.16 Number of iterations and time required to solve problems 1, 2 and 3 by non linear conjugate gradient method and evaluating the Jacobian every 10-iterations

($||R_i||_2 < \varepsilon$, $\varepsilon = 10^{-5}$, for convergence checking)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	65	0.53907	86	0.50298	21	0.12765
100	67	0.73906	89	0.66530	21	0.15416
400	97	7.00430	121	7.02507	37	1.73583
900	113	39.61076	149	21.11819	37	33.27519
1600	129	68.30016	158	90.02504	47	77.87178

Table 4.17 Number of iterations and time required to solve problems 1, 2 and 3 by linear conjugate gradient method ($||R_i||_\infty < \varepsilon$, $\varepsilon = 10^{-5}$, for convergence checking)

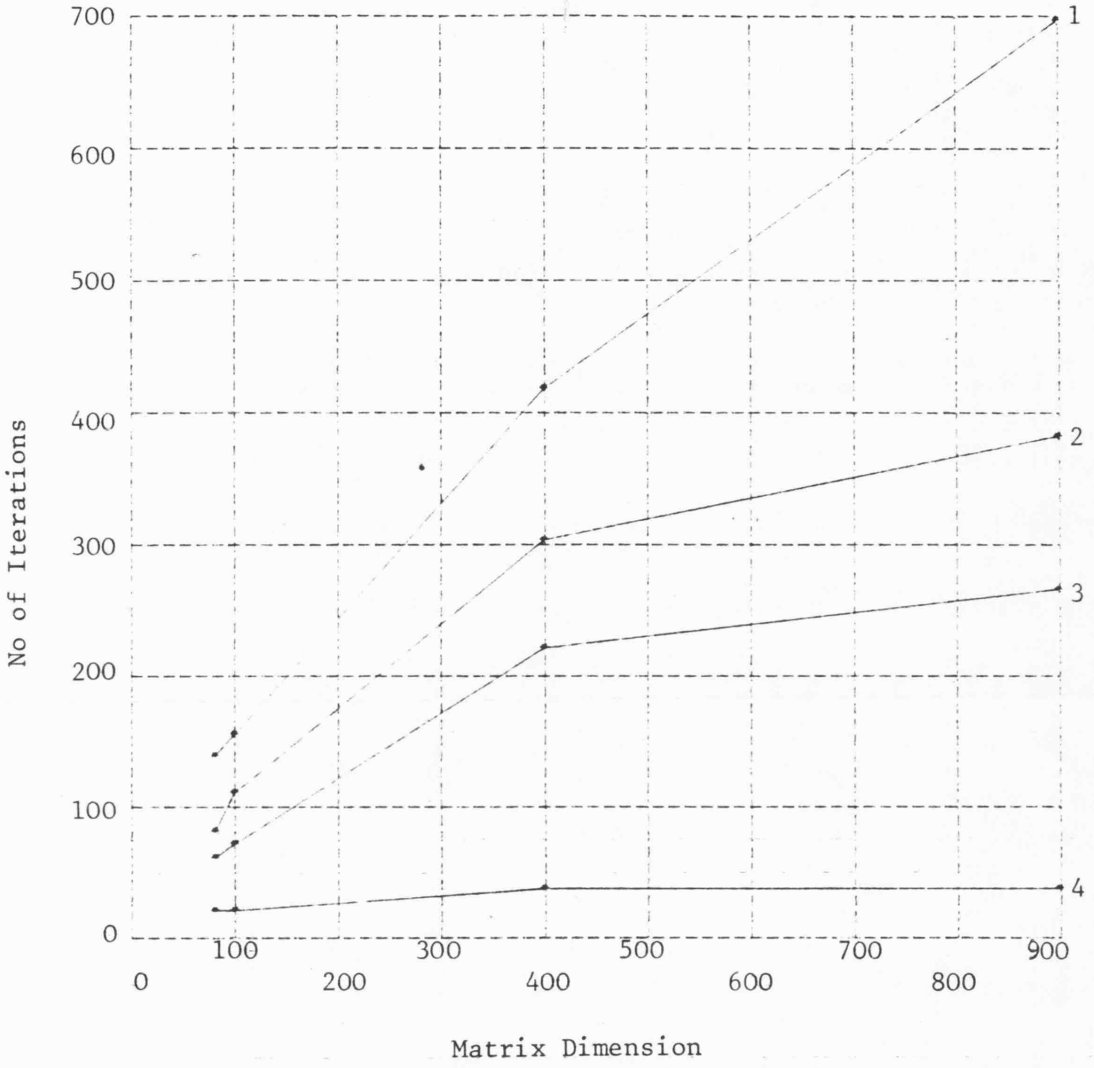


Fig. 4.10

The variation in the number of iterations to solve problem 1 by:

1. Non linear conjugate gradient method (evaluating the Jacobian every iteration)
2. Non linear conjugate gradient method (evaluating the Jacobian every 10 iterations)
3. Non linear conjugate gradient method (evaluating the Jacobian every 5 iterations)
4. Linear conjugate gradient method

   (with the convergence criteria $||x_{i+1} - x_i||_\infty < 10^{-5}$)

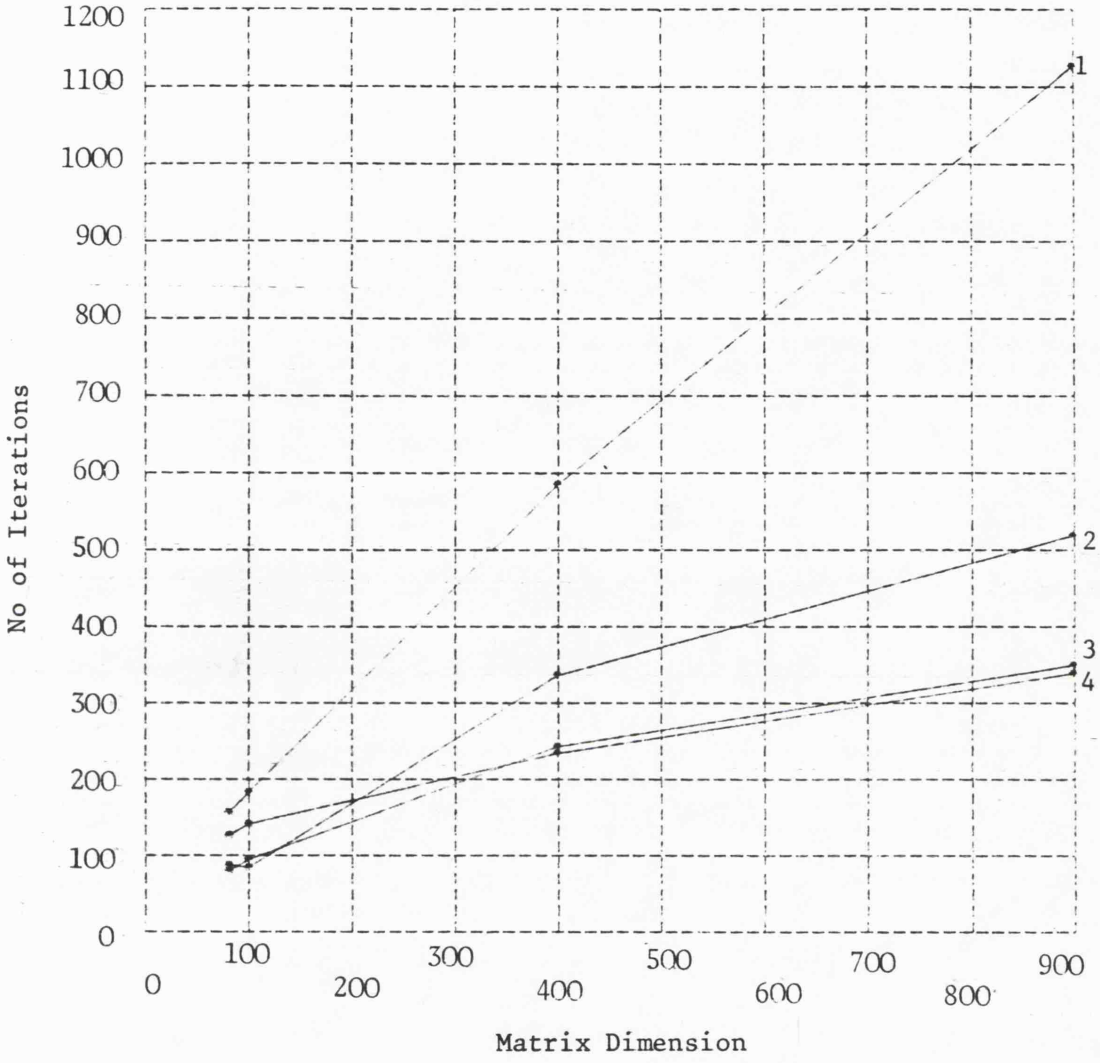


Fig. 4.11

The variation in the number of iterations to solve problem 2 by:

1. Non linear conjugate gradient method (evaluating the Jacobian every iteration)
2. Non linear conjugate gradient method (evaluating the Jacobian every 10-iterations)
3. Non linear conjugate gradient method (evaluating the Jacobian every 5-iterations)
4. Linear conjugate gradient method (with the convergence criterion $||x_{i+1} - x_i||_\infty < 10^{-5}$)

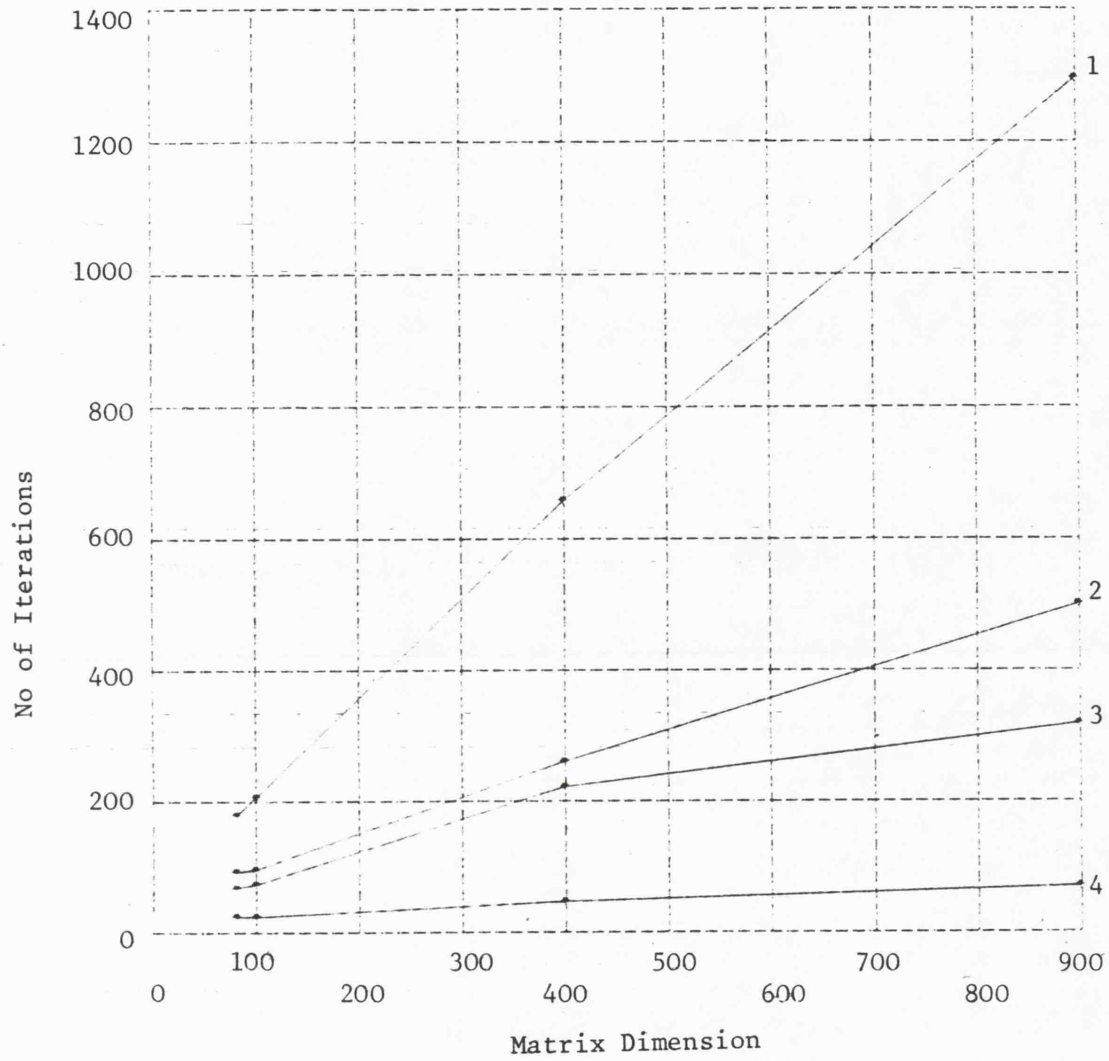


Fig. 4.12

The variation in the number of iterations to solve problem 3 by:

1. Non linear conjugate gradient method (evaluating the Jacobian every iteration)
2. Non linear conjugate gradient method (evaluating the Jacobian every 10-iterations)
3. Non linear conjugate gradient method (evaluating the Jacobian every 5-iterations)
4. Linear conjugate gradient method

   (with the convergence criteria $||x_{i+1} - x_i||_\infty < 10^{-5}$)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	11	0.17378	10	0.10355	2	0.01854
100	12	0.23335	10	0.12659	2	0.02268
400	20	1.56277	19	0.96593	2	0.08935
900	26	4.58600	27	3.10770	2	0.20090
1600	32	10.06600	35	7.23430	2	0.36180

Table 4.18 Number of iterations and time required to solve problems 1, 2 and 3 by preconditioned conjugate gradient method (factorizing the Jacobian every iteration)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	11	0.14193	10	0.08471	2	0.01585
100	12	0.18817	10	0.10507	2	0.01946
400	20	1.24772	19	0.75867	2	0.07632
900	26	3.65820	26	2.35587	2	0.16867
1600	34	8.243160	33	5.22852	2	0.30371

Table 4.19 Number of iterations and time required to solve problems 1, 2 and 3 by preconditioned conjugate gradient method (factorizing the Jacobian every 5-iterations)

Problem Dimension	Problem 1		Problem 2		Problem 3	
	No of iter	Time(sec)	No of iter	Time(sec)	No of iter	Time(sec)
81	11	0.13694	11	0.09044	2	0.01292
100	12	0.18300	11	0.11008	2	0.01553
400	20	1.20052	19	0.72995	2	0.06171
900	22	2.97480	26	2.27281	2	0.13794
1600	26	6.21631	33	5.11792	2	0.24722

Table 4.20 Number of iterations and time required to solve problems 1, 2 and 3 by preconditioned conjugate gradient method (factorizing the Jacobian every 10-iterations)

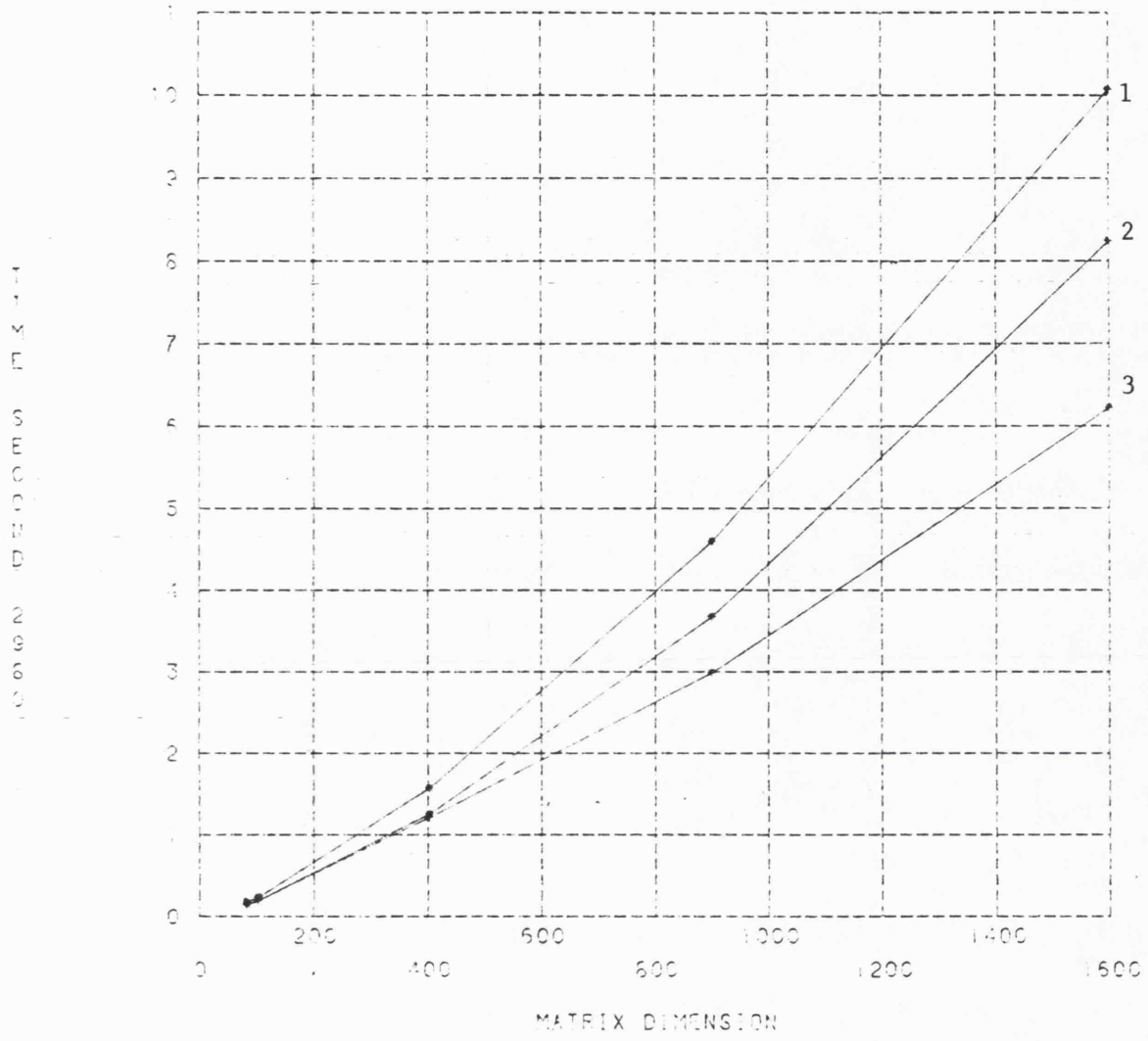


Fig. 4.13

Time required to solve problem 1 by preconditioned non linear conjugate gradient method, factorizing the Jacobian at:

1. Every iteration
2. Every 5-iterations
3. Every 10-iterations

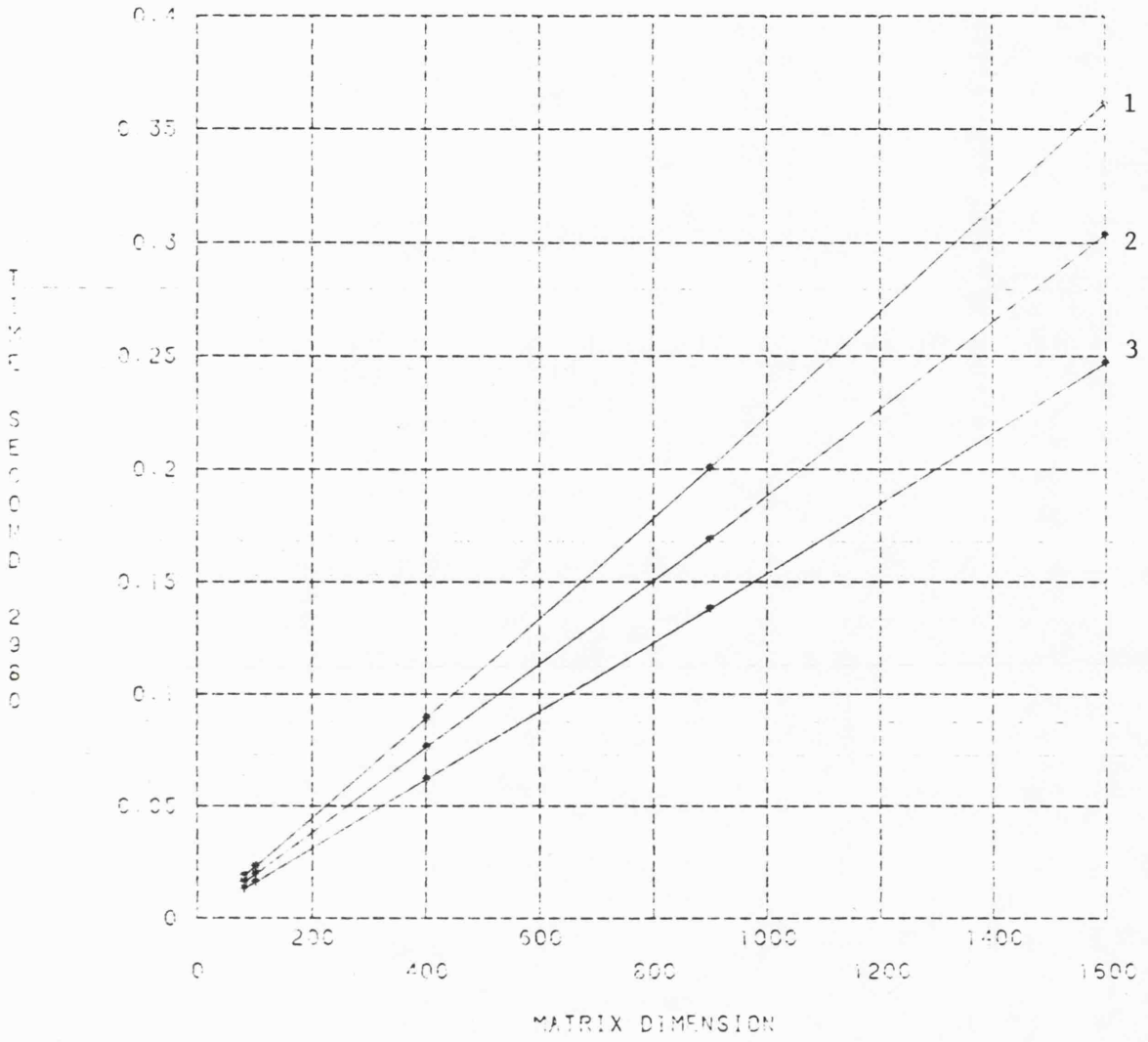


Fig. 4.15

Time required to solve problem 3 by preconditioned non linear conjugate gradient method, factorizing the Jacobian at:

1. Every iteration
2. Every 5-iterations
3. Every 10-iterations

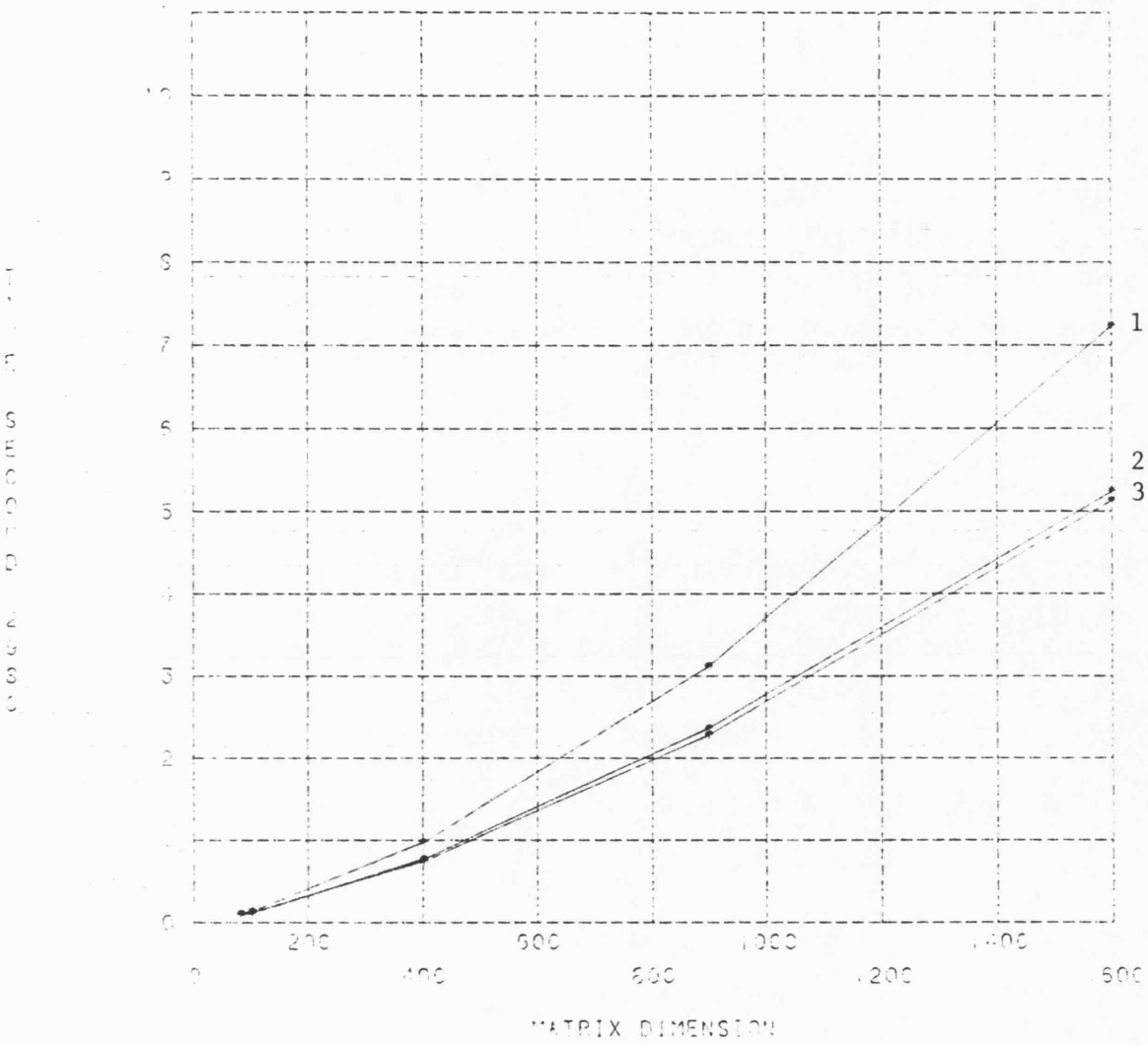


Fig. 4.14

Time required to solve problem2 by preconditioned non linear conjugate gradient method, factorizing the Jacobian at:

1. Every iteration
2. Every 5-iterations
3. Every 10-iterations

With reference to (H8), problems 1, 2 and 3 are solved by the following iterative techniques:

1. Non linear point Gauss Seidel method (NPGS)
2. Non linear point S.O.R. method (NPSOR)
3. Non linear block Gauss Seidel method (NBGS)
4. Non linear block S.O.R. method (NPSOR)

To facilitate the comparison with the results of Hageman and Porsching, we used the same convergence criteria to solve problems 1, 2 and 3 using non linearized S.I.P.

The numerical results from the application of NPGS, NPSOR, NBGS, NBSOR and nonlinearized S.I.P. are listed in Table 4.21, which shows that using nonlinearized S.I.P. required less iterations than Hageman and Porsching's results.

Problem	NPGS	NPSOR	NBGS	NBSOR	Nonlinearized S.I.P.
1	401	83	218	74	43
2		87	208	69	38
3	487	98	264	75	45

Table 4.21

The number of iterations to solve problems 1, 2 and 3 with h = 0.05

CHAPTER 5 Bidiagonalization Method

5.1 Introduction

In Chapter 3 we studied the numerical solution of a system of non-linear equations by using "Linear and Non Linear Conjugate Method" and "Preconditioned Non Linear Conjugate Gradient Method".

As has been explained before the non-linear conjugate gradient method is a generalization of the linear conjugate gradient method, both having the following attractive properties, when used as iterative procedures.

1) No requirement an estimation of parameters
2) Advantage taken of the distribution of eigenvalues of the iteration operator.
3) Fewer restrictions placed on the matrix A for optimal behaviour than in the case with other methods such as successive over relaxation.

But the main restriction in solving the system

$$Ax = b \qquad 5.1.1$$

(where A is an m x n - matrix, $m \geq n$, and an m-vector b) by use of "Conjugate Gradient Method" or "Preconditioned Conjugate Gradient Method" is that conjugate gradient methods are not directly applicable if A is non-symmetric.

We can avoid this problem by solving the normal equations

$$A^TAx = A^Tb \qquad 5.1.2$$

If the matrix A is of rank (A) = n, A^TA is non-singular, and this implies that A^TA is positive definite and we can use the conjugate gradient method to solve 5.1.2 in a least square sense (see (A7), (B6), (B7), (D5), (G5),

(P5)and (P7)). One of the techniques used to solve 5.1.1 is the "Linear Least Squares", i.e. we determine r so that:

$$r + Ax = b, \quad A^T r = 0 \qquad 5.1.3$$

This is equivalent to solving the symmetric indefinite system of linear equations

$$\tilde{\tilde{A}}\tilde{x} = \tilde{b} \qquad 5.1.4$$

where

$$\tilde{A} = \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}, \quad \tilde{b} = \begin{pmatrix} b \\ 0 \end{pmatrix} \text{ and } \tilde{x} = \begin{pmatrix} r \\ x \end{pmatrix}$$

It is clear that A is (m + n) x (m + n), and x and b are vectors of length (m + n), see (L7) and (P4).

P. Concus and G.H. Golub in 1975 (P4) proposed a technique to solve the system 5.1.1.

The technique is summarised by defining a symmetric part derived from the matrix A as $M = M^T = (A + A^T)/2$, where M is called the symmetric part of A, and $N = -N^T = -(A - A^T)/2$ is the negative of its skew part.

The scheme of iteration is as follows

$$x^{k+1} = x^{k-1} + \omega_{k+1}(\alpha_k z^k + x^k - x^{(k-1)})$$

$$Mz^{(k)} = r^{(k)}$$

$$r^{(k)} = b - (M - N)x^k = b - Ax^k$$

Implicitly solving a system of linear equations by any variational method e.g. conjugate gradients includes the estimation of the eigenvalues of a certain matrix.

Such a variational method to evaluate eigenvalues of a certain matrix concentrates on reducing the main matrix to an equivalent form such as a tridiagonal matrix, where eigenvalues can easily be calculated. Such a transformation can be performed by a sequence of transformations which for their performance require the calculation of orthogonal vectors generated throughout the iterative process.

Lanczos (1950) (L1) proposed such a technique to find the eigenvalues of any matrix by reducing the matrix to tridiagonal form, the Lanczos process is as follows:

Choose $v_1 \neq 0$ and $\beta_1 = 0$

$$\alpha_i = v_i^T A v_i / v_i^T v_i \qquad 5.1.6$$

$$\alpha_{i+1} v_{i+1} = A v_i - \alpha_i v_i - \beta_i v_{i-1}$$

(α_{i+1} is a normalizing factor)

$$\beta_{i+1} = v_i^T A v_{i+1} / v_i^T v_i = \alpha_{i+1} v_{i+1}^T v_{i+1} / v_i^T v_i$$

The vectors v_i generated by the Lanczos algorithm are orthogonal so that the resulting matrix form is given by

$$AV = VT \qquad 5.1.7$$

T is a tridiagonal and every eigenvalue of T is also an eigenvalue of A.

But in the computation of the Lanczos' process, cancallation in the vector subtraction step leads to loss of orthogonality in the vectors $v_1, v_2, \ldots$ and the process usually does not finitely terminate. This behaviour led Lanczos to suggest re-orthogonalization and to the widespread disregard of the method in its natural state.

Paige's comment in 1972 (P3) about the Lanczos algorithm was:

> "Although others have suspected that the Lanczos process could still be useful, no-one seemed to be aware that the most popular algorithm does not necessarily converge, and that even when it does, it gives poor results for those eigenvalues".

In 1965 Golub and Khan suggested a numerically stable and fairly fast method for reducing a general matrix A to bidiagonal form in such a way that eigenvalues of A are the same as those of the bidiagonal form.

This method is related to the Lanczos method of minimized iterations for tridiagonalizing asymmetric matrix, as we will see in the next section and like that method is ideally suited for large sparse matrices.

In section 3 is given a full description for the bidiagonalization technique which is based on the Golub and Khan technique to solve the system

$$Ax = b \qquad 5.1.8$$

where A is an m x n matrix and b is an m-vector.

In section 4 we propose a reduction in the number of iterations required to solve 5.1.8 by using Stone factorization (for symmetric and non-symmetric matrices, see (S15)).

5.2 The bidiagonalization algorithm

The bidiagonalization algorith suggested by Golub and Khan for bidiagonalization has been described in detail by Paige (P4). Here we will give just the main points.

For a given m x n matrix A, $m \geq n$, and an initial vector u_1, such that $||u_1|| = 1$, the method generates m-dimensional vectors $u_1, u_2, \ldots$ and n-dimensional vector $v_1, v_2, \ldots$ such that for $i = 1,2,\ldots$

$$\alpha_i v_i = A^T u_i - \beta_i v_{i-1}, \qquad \beta_1 v_0 = 0$$

$$\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i \qquad 5.2.1$$

where α_i, β_{i+1} are real scalars chosen to be non-negative and such that, $||u_{i+1}|| = ||v_i|| = 1$, for non-zero α_i and β_{i+1} where $i = 1,2,\ldots,k$ then 5.2.1 is fully defined for k-steps, and 5.2.1 may be re-written as

$$A^T U = V L^T, \qquad AV = UL + \beta_{k+1} u_{k+1} e_k^T \qquad 5.2.2$$

where $U = [u_1, u_2, \ldots, u_k]$, $V = [v_1, v_2, \ldots, v_k]$ and

$$L = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{bmatrix} \qquad 5.2.3$$

where e_k is the k-th unit vector so that

$$U^T AV = U^T UL + \beta_{k+1} U^T u_{k+1} e_k^T = L V^T V \qquad 5.2.4$$

If α_{k+1} is also a non-zero and $i = k+1$, with $\tilde{U} = [U, u_{k+1}]$ $\tilde{L} = [L^T, \beta_{k+1} e_k]^T$, then

$$A^T \tilde{U} = V \tilde{L}^T + \alpha_{k+1} v_{k+1} e_{k+1}^T, \qquad AV = \tilde{U}\tilde{L} \qquad 5.2.5$$

Therefore

$$V^T A^T \tilde{U} = V^T V \tilde{L}^T + \alpha_{k+1} V^T v_{k+1} e_{k+1}^T = \tilde{L}^T \tilde{U}^T \tilde{U} \qquad 5.2.6$$

By induction, we have that

$$U^T U = V^T V = I_k \qquad 5.2.7$$

(for the proof see Paige (P4))

The orthogonality of the generated vectors ensures that the process will be limited for some k, given by

$$k \leq \min \{m,n\} \qquad 5.2.8$$

such that either $\beta_{k+1}u_{k+1} = 0$ in 5.2.4 or $\alpha_{k+1}v_{k+1} = 0$ in 5.2.5.

Therefore we have two possible final forms of the Bidiagonalization algorithm

1. When L is a k x k-matrix

$$A^TU = VL^T, \quad AV = UL, \quad U^TU = V^TV = I_k \qquad 5.2.9$$

2. When L is a (k + 1) x k-matrix

$$A^T\tilde{U} = V\tilde{L}^T, \quad AV = \tilde{U}\tilde{L}, \quad \tilde{U}^T\tilde{U} = I_{k+1}, \quad V^TV = I_k$$

(For more about the Bidiagonalization technique see Golub and Khan (G6) and Paige (P4)).

The matrix form of the bidiagonalization algorithm is equivalent to the application of the Lanczo's algorithm on the matrix

$$B = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$$

such that the orthogonal vector is given by

$$w = (w_1,w_2,\ldots,w_{2k}) = \begin{pmatrix} u_1 & 0 & u_2 & 0 & \ldots & u_k & 0 \\ 0 & v_1 & 0 & v_2 & \ldots & 0 & v_k \end{pmatrix}$$

and the equivalent tridiagonal form is as follows

$$T = \begin{bmatrix} 0 & \alpha_1 & & & & & \\ \alpha_1 & 0 & \beta_2 & & & & \\ & \beta_2 & 0 & \alpha_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \alpha_k & 0 & \beta_k & \end{bmatrix}$$

Therefore 5.2.9 is equivalent to

$$BW = WT + B_{k+1}\, w_{2k+1}\, e_{2k}^T \qquad W^TW = I_{2k} \qquad W^Tw_{2k+1} = 0$$

which is the result of applying 2k-steps of the Lanczos process to B, using as initial vector w_2, the first column of W.

5.3 The bidiagonalization technique to solve the linear least squares problem

For an m x n matrix A and an m-vector b, the problem is

$$\text{minimize } ||Ax - b|| \qquad 5.3.1$$

Or equivalently find x and r such that

$$r + Ax = b, \quad A^Tr = 0 \qquad 5.3.2$$

Any such x is called "a Least Squares" solution, and the x which also minimizes $||x||$ is called the minimum least squares solution. (The minimum least squares solution is the unique solution orthogonal to N(A)), so x will have the form $x = A^Ty$. Thus if y is any solution of

$$A^T A\, A^Ty = A^Tb \qquad 5.3.3$$

Then $x = A^Ty$ is the minimum least squares solution.

Paige (P4) defined the convenient choice of u in the bidiagonalization algorithm in such a way as to produce an iterative solution vector x, such that such a representation $x = A^Ty$ is possible.

Equations in the form 5.3.1 or 5.3.2 can be separated into two possible classes with corresponding slightly different methods of solution. First if the linear equation Ax = b is "compatible" (i.e. if r = 0) then $b \in R(A)$ and so $u_1 \in R(A)$, (see P4).

The second class is when the linear system is "Incompatible", i.e. $(r \neq 0)$(see (P4)).

Finally the algorithm for compatible and incompatible system is as follows

Algorithm 5.3.1

1. $\tau_0 = 1$, $\omega_0 = 0$, $\eta_0 = -1$, $vz = 0$, $v\omega = 0$

   $\beta_1 u_1 = b$

   $\alpha_1 v_1 = A^T u_1$

   set $i = 1$

2. Iteration i

   $\eta_i = -\eta_{i-1}\ \beta_i/\alpha_i$ $\qquad vz = vz + \eta_i v_i$

   $\omega_i = (\tau_{i-1} - \beta_i \omega_{i-1})/\alpha_i$ $\qquad v\omega = v\omega + \omega_i v_i$

   $\beta_{i+1} u_{i+1} = Av_i - \alpha_i u_i$

   If $\beta_{i+1} = 0$, then (solution = vz, residual = 0, STOP)

   $\tau_i = -\tau_{i-1}\ \alpha_i/\beta_{i+1}$

   $\alpha_i v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i$

   If $\alpha_{i+1} = 0$, then

   ($\gamma = \beta_{i+1}\eta_i/(\beta_{i+1}\omega_i - \tau_i)$, solution = $vz - \gamma v\omega$, STOP)

3. Set $i = i + 1$, go to 2

Note: From the numerical applications of algorithm 5.3.1 to solve a system of non linear equations, we found it is more reliable to use η_{i+1} as a convergence criteria rather than β_{i+1}.

Remark: With some problems, the stopping criteria appearing in the above algorithm would be almost useless, and in fact, it could happen that no α_i or β_i is even small, so we use instead η_i and ω_i as a stopping criteria.

Theorem 5.3.1

At the i-th iteration, we have the following relations

1(a) $A^TAv_i \in \{v_1,v_2,\ldots,v_{i+1}\}$ 1(b) $AA^Tu_i \in \{u_1,u_2,\ldots,u_{i+1}\}$

2 $A^Tr_i \in \{v_1,v_2,\ldots,v_{i+1}\}$

3(a) $Av_i \in \{u_1,u_2,\ldots,u_{i+1}\}$ 3(b) $A^Tu_i \in \{v_1,v_2,\ldots,v_{i+1}\}$

4 $r_i \in \{u_1,u_2,\ldots,u_{i+1}\}$

5 $\{u_1,u_2,\ldots,u_{i+1}\} = \{u_1,AA^Tu_1,\ldots,(AA^T)^{i+1}u_1\}$

$= \{r_0, AA^Tr_0,\ldots,(AA^T)^{i+1}r_0\}$

6 $\{v_1,v_2,\ldots,v_{i+1}\} = \{v_1,A^TAv_1,\ldots,(A^TA)^{i+1}v_1\}$

Proof

To prove 1(a)

From algorithm 4.3.1 we have

$$\beta_{i+1}u_{i+1} = Av_i - \alpha_iu_i \qquad 5.3.4$$

$$\alpha_{i+1}v_{i+1} = A^Tu_{i+1} - \beta_{i+1}v_i \qquad 5.3.5$$

Therefore

$$\alpha_{i+1}v_{i+1} = \frac{1}{\beta_{i+1}}A^TAv_i - \frac{\alpha_i}{\beta_{i+1}}A^Tu_i - \beta_{i+1}v_i$$

$$= \frac{1}{\beta_{i+1}}A^TAv_i - \frac{\alpha_i}{\beta_{i+1}}\alpha_iv_i - \frac{\alpha_i\beta_i}{\beta_{i+1}}v_{i-1} - \beta_{i+1}v_i$$

Hence we conclude that

$$\beta_{i+1}\alpha_{i+1}v_{i+1} = A^TAv_i - (\alpha_i^2 + \beta_{i+1}^2)v_i - \beta_i\alpha_iv_{i-1}$$

The proof of 1(b) is similar to 1(a).

To prove 2

The residual at the i-th iteration is given by

$$r_i = -\beta_{i+1}\eta_i u_{i+1}$$

Hence

$$A^T r_i = -\beta_{i+1}\eta_i A^T u_{i+1}$$

and from algorithm 5.3.1 we have

$$A^T u_{i+1} = \alpha_{i+1} v_{i+1} + \beta_{i+1} v_i$$

Therefore

$$A^T r_i = -\beta_{i+1}\eta_i(\alpha_{i+1} v_{i+1} + \beta_{i+1} v_i)$$

$$= -\beta_{i+1}\eta_i\, \alpha_{i+1} v_{i+1} - \beta_{i+1}^2 \eta_i v_i$$

Therefore $A^T r_i \in \{v_1, v_2, \ldots, v_{i+1}\}$

3(a) and 3(b) follows from the algorithm 5.3.1 directly.

To prove 4

$$r_i = -\beta_{i+1}\eta_i u_{i+1}$$

Therefore $r_i \in \{u_1, u_2, \ldots, u_{i+1}\}$

To prove 5 (The proof of 6 is similar to 5)

$$\{u_1, u_2, \ldots, u_{i+1}\} = \{u_1, AA^T u_1, \ldots, (AA^T)^{i+1} u_1\}$$

The proof is by induction, it is clear that for i = 1 the relation is satisfied.

Therefore for i = 2, from the algorithm we have

$$u_2 = Av_1 - u_1$$

$$= \frac{1}{\alpha_1} AA^T u_1 - u_1$$

Therefore $\{u_1, u_2\} = \{u_1, AA^T u_1\}$

Assume that the relation is satisfied for $i = k - 1$

i.e. $\{u_1, u_2, \ldots, u_k\} = \{u_1, AA^T u_1, \ldots, (AA^T)^k u_1\}$

For $i = k + 1$

$$\beta_{k+1} u_{k+1} = Av_k - \alpha_k u_k$$

Therefore by 3(a) it follows that

$$\{u_1, u_2, \ldots, u_{k+1}\} = \{u_1, AA^T u_1, \ldots, (AA^T)^{k+1} u_1\}$$

since $r_i = - \beta_{i+1} \eta_i u_{i+1}$

Therefore $\{u_1, u_2, \ldots, u_{i+1}\} = \{u_1, AA^T u_1, \ldots, (AA^T)^{i+1} u_1\}$

$$= \{r_0, AA^T r_0, \ldots, (AA^T)^{i+1} r_0\}$$

Theorem 5.3.2

At the i-th step of iteration we have

$$E(r_i) \leq P_i \, E(r_0)$$

(where $E(r_i)$ is the error in the residual get it at the i-th iteration)

$P_i = I - AA^T P_{i-1}(AA^T)$; P_i is a polynomial of degree i in AA^T

Proof

From the bidiagonalization algorithm we have

$$x_i = x_{i-1} + \eta_i v_i = x_0 + \sum_{j=1}^{i} \eta_j v_j$$

$$r_i = r_o - \sum_{j=1}^{i} \eta_j A v_j$$

By theorem 5.3.1 and algotithm 5.3.1

$$r_i = r_0 - \sum_{j=1}^{i} \eta_j (AA^T)^j r_0$$

$$r_i = \{I - P_i(AA^T)\}\, r_0 = \{I - AA^T P_{i-1}(AA^T)\}\, r_0$$

$$R_i\,(AA^T) = I - AA^T P_{i-1}(AA^T)$$

R_i is a polynomial of degree i, s.t. $R_i(0) = I$

$$E(r_i) = \min_{R_i} ||R_i(AA^T) r_0||_2^2 = \min(R_i(AA^T) r_0,\ R_i(AA^T) r_0)$$

Since AA^T is symmetric, therefore it has m-orthonormal eigenvector $\{w_j\}_{j=1}^m$.

Hence

$AA^T w_j = \lambda_j w_j$, where $\{\lambda_j\}_{j=1}^m$ are the eigenvalues of AA^T

$$r_0 = \sum_{j=1}^{m} s_j w_j \qquad \{s_j\} \text{ are any constants}$$

$$R_i(AA^T) r_0 = \sum_{j=1}^{m} s_j R_j(AA^T) w_j = \sum_{j=1}^{m} s_j R_i(\lambda_j) w_j$$

$$E(r_i) = \min_{R_i} \Big(\sum_{j=1}^{m} s_j R_i(\lambda_j) w_j,\ \sum_{j=1}^{m} s_j R_i(\lambda_j) w_j \Big)$$

$$= \min_{R_i} \sum_{j=1}^{m} s_j^2 R_i^2(\lambda_j) \qquad \text{(by the orthonormality of } w_j\text{)}$$

$$\leq \min_{R_i} \Big(\max_{1 \leq j \leq m} |R_i(\lambda_j)| \Big)^2 \sum_{j=1}^{m} s_j^2$$

$$= \min_{R_i} \Big(\max_{1 \leq j \leq m} |R_i(\lambda_j)| \Big)^2 \Big(\sum_{j=1}^{m} s_j w_j,\ \sum_{j=1}^{m} s_j w_j \Big)$$

$$= \min_{R_i} \Big(\max_{1 \leq j \leq m} |R_i(\lambda_j)| \Big)^2 (r_0, r_0)$$

Let $P_i = \min_{R_i} \Big(\max_{1 \leq j \leq m} |R_i(\lambda_j)| \Big)^2$

Therefore $E(r_i) \leq P_i\, E(r_0)$

Theorem 5.3.3

If r_0 has non-zero components for $k \leq n$ (for k-eigenvalues of AA^T) corresponding to only $m \leq k$ distinct eigenvalues of AA^T then the method converges in at most m-iterations to a unique solution of $Ax = b$.

Proof

Assume that r_0 has non zero components only along the eigenvectors $w_1, w_2, \ldots, w_k$ of AA^T, then

$$r_0 = \sum_{j=1}^{k} s_j w_j$$

Therefore

$$E(r_i) \leq P_i E(r_0) \qquad P_i = \min_{R_i} \max_{1 \leq j \leq k} |R_i(\lambda_j)|$$

Therefore for $\lambda_1, \lambda_2, \ldots, \lambda_m$ distinct eigenvalues of AA^T from the k-eigenvalues.

Therefore $P_i = \min_{R_i} \max_{1 \leq j \leq m} |R_i(\lambda_j)|$

Therefore the polynomial $R_m(y) = \prod_{j=1}^{m} \left(\frac{y - \lambda_j}{\lambda_j}\right)$ satisfies $R_m(\lambda_j) = 0$.

Therefore $E(r_i) = 0$, for $i \leq m$.

Theorem 5.3.2 and 5.3.3 do not give a minimum bound for the polynomial $R_i(AA^T)$ and we cannot evaluate P_i as well. But by using the normalized chybshev polynomial we can find the upper bound on P_i.

Since AA^T is a positive definite matrix, let $[a,b]$ be an interval containing all the eigenvalues of AA^T, $0 < a < b$.

To obtain a bound on P_i, choose $R_i(\lambda)$ to be the unique polynomial in $R_i(\lambda)$, that minimizes the deviation from zero on $[a,b]$, and the normalized chybshev polynomial on $[a,b]$

$$R_i(\lambda) = \frac{T_i(\frac{-2\lambda + a + b}{b - a})}{T_i(\frac{b + a}{b - a})}$$

where $T_j(x) = \cos(j \cos^{-1} x)$ is the j-th chybshev polynomial in x.

Therefore

$$P_i \leq 2(\frac{1 - \sqrt{p}}{1 + \sqrt{p}})^i \quad , \quad i \geq 0$$

where $p = \frac{1}{k(AA^T)}$, $k(AA^T)$ is the condition number of AA^T.

Theorem 5.3.4

At the i-th iteration, we have

$$E(r_i) \leq 2(\frac{1 - \sqrt{p}}{1 + \sqrt{p}})^i E(r_0)$$

Theorem 5.3.5

If AA^T is positive definite matrix, then

1. O(m logm) iterations are required to reduce the 2-norm of the initial error by a factor $m^{-\alpha}$, $\alpha > 0$.
2. The rate of convergence R is given by $R \sim \pi h$.

Proof

By theorem 4

$$E(r_i) \leq 2\sqrt{k(AA^T)} \quad (\frac{1 - \sqrt{p}}{1 + \sqrt{p}}) E(r_0)$$

where $p = c_1^2/m^2$, $\sqrt{k(AA^T)} = c_2 m^q$, and c_1, c_2, $q > 0$, are constants and are independent of m.

Suppose that it takes j-iterations to reduce the error bound by factor of $m^{-\alpha}$, $\alpha > 0$, then

$$2c_2 m^q (\frac{1 - c_1/m}{1 + c_1/m})^j \leq \frac{1}{m^\alpha}$$

$$\text{Or} \quad 2c_2 m^{q+\alpha} \leq \left(\frac{1 + c_1/m}{1 - c_1/m}\right)^j$$

Because of that the log function is monotically increasing on $[1,\infty)$.

Therefore

$$j \geq (c_3 \log m)/\log \left(\frac{1 + c_1/m}{1 - c_1/m}\right)$$

where c_3 is a constant independent of m

and since $c_1 < m$, therefore

$$\log \left(\frac{1 + c_1/m}{1 - c_1/m}\right) = \frac{2c_1}{m}$$

Hence

O(m logm) iterations are sufficient.

Let $p = \pi^2 h^2/4$, where h is the mesh size, $h = \frac{1}{m + 1}$

Then

$$\frac{1 - \sqrt{p}}{1 + \sqrt{p}} = \frac{1 - \pi h/2}{1 + \pi h/2} \sim 1 - \pi h, \text{ as } h \longrightarrow 0$$

And the rate of convergence R is given by

$$R = -\log (1 - \pi h) \sim \pi h \quad \text{as } h \longrightarrow 0$$

The work required to solve any system of linear equations (with a symmetric and positive definite coefficient matrix) by the bidiagonalization technique is twice the amount required to solve such a system by using the conjugate gradient technique.

As has been explained in chapter 3 an improvement (acceleration) to the conjugate gradient has been proposed by using an approximate factorization for the associated matrix. With respect to the bidiagonalization technique, there are two ways to improve the convergence, by using an approximate factorization C of the matrix A in 5.1.1, in the next section we introduce

a full description of what we call "post-preconditioned bidiagonalization technique" and "pre-preconditioned bidiagonalization technique".

5.4 The preconditioning bidiagonalization technique

As has been explained before the rate of convergence of the bidiagonalization technique depends on $k(A^TA)$, which is therefore a relatively expensive way to solve a system of linear equations. 5.1.1

Let C be any approximate factorization for the matrix A, then there are two least squares problems generated from 5.1.1 by associating the approximate factorizations C for the matrix A.

5.4.1 Post-preconditioning bidiagonalization technique

The linear least squares method to solve 5.1.1 is

$$\text{minimize } ||AC^{-1}z - b|| \; ; \quad Cx = z \qquad 5.4.1$$

were C is any approximate factorization for the n x n matrix A, or to find z(x) and r such that $r + AC^{-1}z = b, C^{-T}A^Tr = 0$. Any such z(x) is called "least square solution".

The minimum least square solution is the unique solution orthogonal to $N(AC^{-1})$, and it will have the form $Cx = z = C^{-T}A^Ty$.

So, if y is a solution of $C^{-T}A^TAC^{-1}C^{-T}A^Ty = C^{-T}A^Tb$

then $z = Cx = C^{-T}A^Ty$ is the minimum least square solution.

Suppose $AC^{-1}V = UL \qquad Cx = z = Vy, \quad x = C^{-1}z = C^{-1}Vy$

and since $r^TAC^{-1}V = r^TUL = 0$, L is non-singular, $U^Tr = 0$

$$b = r + AC^{-1}z = r + AC^{-1}Vy = r + ULy$$

Therefore x and y can be found from

$$Ly = U^T b$$

Hence we conclude that $z = Vy$ and $x = C^{-1}Vy$

Now to show that for a special choice of u_1, such a representation of x is possible, when the linear system is compatible (i.e. $r = 0$).

Therefore with respect to the equation $AC^{-1}z = b$, $b \in R(A)$

Let $u_1 = b/\beta_1$, $\beta_1 = ||b||_2$

Therefore $u_1 \in R(AC^{-1})$ and $AC^{-1}z = b = \beta_1 u_1$

$$C^{-T}A^T u = VL^T \text{ and } AC^{-1}V = UL$$

where $UU^T = VV^T = I$

Let $z = Cx = Vy + w$; $x = C^{-1}z = C^{-1}Vy + C^{-1}w$, where $V^T w = 0$

Then

$$Ax = AC^{-1}z = ULy + AC^{-1}w = \beta_1 u_1$$

$$ULy + AC^{-1}w = \beta_1 u_1$$

which implies that

$$Ly + U^T AC^{-1}w = \beta_1 u_1$$

Therefore

$$Ly + LV^T w = \beta_1 u_1$$

But since $V^T w = 0$, therefore

$Ly = \beta_1 e_1$, and thus $AC^{-1}w = 0$ and $z = Vy$ where x is given by $x = Cz$.

The elements of y are given by $Ly = \beta_1 e_1$,

i.e. $\eta_1 = \beta_1/\alpha_1$ and $\eta_{i+1} = -(\beta_{i+1}/\alpha_{i+1})\eta_i$.

The second possibility to be examined now is the one when the linear system is "uncompatible" (i.e. $r \neq 0$), so that $u_1 = b/\beta_1 \in R(A)$, therefore the problem is to solve:

$$r + AC^{-1}z = b = \beta u_1 \qquad A^T r = 0,\ Cx = z \qquad 5.4.2$$

where $C^{-T}A^T\tilde{U} = V\tilde{L}^T$, $AC^{-1}V = \tilde{U}\tilde{L}$

where $\tilde{U} = [U, u_{k+1}]$; $\tilde{L} = [L^T, \beta_{k+1}e_k]^T$ (see Paige (P4))

such that $\tilde{U}^T\tilde{U} = I_{k+1}$ $\qquad V^TV = I_k$

Assume that $z = Vy + w$ with $V^Tw = 0$

Therefore $r + \tilde{U}\tilde{L}y + AC^{-1}w = \beta_1 u_1$, but $\tilde{U}^TAC^{-1}w = \tilde{L}V^Tw$

so $\tilde{L}y = \beta_1 e_1 - \tilde{U}^T r$

Hence we conclude that

$$r + AC^{-1}w = \tilde{U}\tilde{U}^T r$$

Thus $w^TC^{-T}A^Tr + w^TC^{-T}A^TAC^{-1}w = ||AC^{-1}w||^2 = w^TC^{-T}A^T\tilde{U}\tilde{U}^Tr = 0$

Thus $AC^{-1}w = 0$, and a smaller solution is obtained by taking $z = Vy$, which implies that the solution vector x is given by $x = C^{-1}Vy$

Therefore the algorithm for both cases, r is close to zero (compatible) and r is not close to zero (incompatible).

Algorithm 5.4.1

The post-preconditioned bidiagonalization method

1. $\tau_0 = 1$, $w_0 = 0$, $\eta_0 = -1$, $vz = 0$, $vw = 0$

   $\beta_1 u_1 = b$

   solve $C^T u_1^* = A^T u_1$ for u_1^* ($u_1 = C^{-T}A^T u_1$)

   $\alpha_1 v_1 = u_1^*$

2. Iteration i

$\eta_i = -\eta_{i-1}\beta_i/\alpha_i$ $\quad vz = vz + \eta_i v_i$

$w_i = (\tau_{i-1} - \beta_i w_{i-1})/\alpha_i$ $\quad vw = vw + w_i v_i$

solve $v_i = Cv_i^*$ for v_i^*

$\beta_{i+1}u_{i+1} = Av_i^* - \alpha_i u_i$

If $\beta_{i+1} = 0$, then (solution = vz, residual = 0, STOP)

$\tau_i = \tau_{i-1}\alpha_i/\beta_{i+1}$

solve $A^T u_{i+1} = C^T u_{i+1}^*$ for u_{i+1}^*

$\alpha_{i+1}v_{i+1} = u^*_{i+1} - \beta_{i+1}v_i$

If $\alpha_{i+1} = 0$, then

($\gamma = \beta_{i+1}\eta_i/(\beta_{i+1}w_i - \tau_i)$, solution = vz - γvw STOP)

3. Otherwise set i = i + 1, and go to 2

Theorem 5.4.1

At the i-th iteration, we have the following

1(a) $C^{-T}A^TAC^{-1}v_i \in \{v_1, v_2, \ldots, v_{i+1}\}$

1(b) $AC^{-1}C^{-T}A^Tu_i \in \{u_1, u_2, \ldots, u_{i+1}\}$

2 $C^{-T}A^Tr_i \in \{v_1, v_2, \ldots, v_{i+1}\}$

3(a) $AC^{-1}v_i \in \{u_1, u_2, \ldots, u_{i+1}\}$

3(b) $C^{-T}A^Tu_i \in \{v_1, v_2, \ldots, v_{i+1}\}$

4 $r_i \in \{u_1, u_2, \ldots, u_{i+1}\}$

5 $\{u_1, u_2, \ldots, u_{i+1}\} = \{u_1, AC^{-1}C^{-T}A^Tu_1, \ldots, (AC^{-1}C^{-T}A^T)^{i+1}u_1\}$

$= \{r_0, AC^{-1}C^{-T}A^Tr_0, \ldots, (AC^{-1}C^{-T}A^T)^{i+1}r_0\}$

6 $\{v_1, v_2, \ldots, v_{i+1}\} = \{v_1, C^{-T}A^TAC^{-1}v_1, \ldots, (C^{-T}A^TAC^{-1})^{i+1}v_1\}$

Proof

To prove 1(a)

$$\beta_{i+1}u_{i+1} = AC^{-1}v_i - \alpha_i u_i$$

$$\alpha_{i+1}v_{i+1} = C^{-T}A^T u_{i+1} - \beta_{i+1}v_i$$

Therefore

$$\alpha_{i+1}v_{i+1} = \frac{1}{\beta_{i+1}} C^{-T}A^T AC^{-1}v_i - \frac{\alpha_i}{\beta_{i+1}} C^{-T}A^T u_i - \beta_{i+1}v_i$$

$$= \frac{1}{\beta_{i+1}} C^{-T}A^T AC^{-1}v_i - \frac{\alpha_i}{\beta_{i+1}} (\alpha_i v_i) - \frac{\beta_i \alpha_i}{\beta_{i+1}} v_{i-1} - \beta_{i+1}v_i$$

Hence we conclude that

$$\beta_{i+1}\alpha_{i+1}v_{i+1} = C^{-T}A^T AC^{-1}v_i - (\alpha_i^2 + \beta_{i+1}^2)v_i - \beta_i \alpha_i v_{i-1}$$

Therefore $C^{-T}A^T AC^{-1}v_i \in \{v_1, v_2, \ldots, v_{i+1}\}$

The proof of 1(b) is similar to 1(a)

To prove 2

The residual at the i-th iteration is given by

$$r_i = -\beta_{i+1}\eta_i u_{i+1}$$

Hence

$$C^{-T}A^T r_i = -\beta_{i+1}\eta_i C^{-T}A^T u_{i+1}$$

From algorithm 5.5.1 we have

$$C^{-T}A^T u_{i+1} = \alpha_{i+1}v_{i+1} + \beta_{i+1}v_i$$

Therefore

$$C^{-T}A^T r_i = -\beta_{i+1}\eta_i(\alpha_{i+1}v_{i+1} + \beta_{i+1}v_i)$$

$$= -\beta_{i+1}\eta_i\alpha_{i+1}v_{i+1} - \beta_{i+1}^2\eta_i v_i$$

Therefore

$$C^{-T}A^{T}r_i \in \{v_1, v_2, \ldots, v_{i+1}\}$$

3(a) and 3(b) follow from the algorithm 4.5.1

To prove 4

Since $r_i = -\beta_{i+1}\eta_i u_{i+1}$

Therefore $r_i \in \{u_1, u_2, \ldots, u_{i+1}\}$

To prove 5, the proof is by induction.

For i = 2, from the algorithm 5.4.1 we have

$$u_2 = AC^{-1}v_1 - u_1$$

$$= \frac{1}{\alpha_1} AC^{-1}C^{-T}A^{T}u_1 - u_1$$

Therefore $\{u_1, u_2\} = \{u_1, AC^{-1}C^{-T}A^{T}u_1\}$

Assume that the relation is satisfied for i = k

i.e. $\{u_1, u_2, \ldots, u_k\} = \{u_1, AC^{-1}C^{-T}A^{T}u_1, \ldots, (AC^{-1}C^{-T}A^{T})^{k}u_1\}$

For i = k + 1

$$\beta_{k+1}u_{k+1} = AC^{-1}v_k - \alpha_k u_k$$

Therefore by 3(a), it follows that

$$\{u_1, u_2, \ldots, u_{k+1}\} = \{u_1, AC^{-1}C^{-T}A^{T}u_1, \ldots, (AC^{-1}C^{-T}A^{T})^{k+1}u_1\}$$

The proof of 6 is similar to 5.

Theorem 5.4.2

At the i-th step of iteration we have

$$E(r_i) \leq P_i E(r_0), \quad P_i = I - AC^{-1}C^{-T}A^{T}P_{i-1}(AC^{-1}C^{-T}A^{T})$$

($E(r_i)$ is the error in the residual at the i-th iteration.)

Proof:

From the post preconditioned bidiagonalization technique we have

$$z_i = z_{i-1} + \eta_i v_i = z_0 + \sum_{j=1}^{i} \eta_j v_j \quad \text{(where } AC^{-1}z = b\text{)}$$

$$r_i = r_0 - \sum_{j=1}^{i} \eta_j AC^{-1} v_j$$

By theorem 5.4.1 and algorithm 5.4.1

$$r_i = r_0 - \sum_{j=1}^{i} \eta_j (AC^{-1}C^{-T}A^T)^j r_0$$

$$r_i = \{I - P_i(AC^{-1}C^{-T}A^T)\}r_0 = \{I - AC^{-1}C^{-T}A^T P_{i-1}(AC^{-1}C^{-T}A^T)\}r_0$$

$$R_i(AC^{-1}C^{-T}A^T) = I - AC^{-1}C^{-T}A^T P_{i-1}(AC^{-1}C^{-T}A^T)$$

where R_i is a polynomial of degree i, such that $R_i(0) = I$

$$E(r_i) = \min_{R_i} ||R_i(AC^{-1}C^{-T}A^T)r_0||_2^2$$

$$= \min_{R_i} (R_i(AC^{-1}C^{-T}A^T)r_0, R_i(AC^{-1}C^{-T}A^T)r_0)$$

Since $AC^{-1}C^{-T}A^T$ is symmetric matrix, therefore it has m-orthonormal eigenvector $\{w_j\}_{j=1}^m$

such that:

$AC^{-1}C^{-T}A^T w_j = \lambda_j w_j$, where $\{\lambda_j\}_{j=1}^m$ are the eigenvalues of $AC^{-1}C^{-T}A^T$

$$r_0 = \sum_{j=1}^{m} s_j w_j, \quad \{s_j\} \text{ are any constants}$$

$$R_i(AC^{-1}C^{-T}A^T)r_0 = \sum_{j=1}^{m} s_j R_i(AC^{-1}C^{-T}A^T)w_j$$

$$= \sum_{j=1}^{m} s_j R_i(\lambda_j)w_j$$

$$E(r_i) = \min_{R_i} \left(\sum_{j=1}^{m} s_j R_i(\lambda_j)w_j, \sum_{j=1}^{m} s_j R_i(\lambda_j)w_j \right)$$

$$= \min_{R_i} \left(\sum_{j=1}^{m} s_j^2 R_i^2(\lambda_j) \right) \quad \text{by the orthonormality of } w_j$$

$$\leq \min_{R_i} \left(\max_{1 \leq j \leq m} |R_i(\lambda_j)| \right)^2 \sum_{j=1}^{m} s_j^2$$

$$= \min_{R_i} \left(\max_{1 \leq j \leq m} |R_i(\lambda_j)| \right)^2 \left(\sum_{j=1}^{m} s_j w_j, \sum_{j=1}^{m} s_j w_j \right)$$

$$= \min_{R_i} \left(\max_{1 \leq j \leq m} |R_i(\lambda_j)| \right)^2 (r_0, r_0)$$

Let $P_i = \min_{R_i} \left(\max_{1 \leq j \leq m} |R_i(\lambda_j)| \right)^2$

Therefore $E(r_i) \leq P_i \, E(r_0)$

Theorem 5.4.3

If r_0 has non-zero components along $k \leq m$, (for k-eigenvalues of $AC^{-1}C^{-T}A^T$) corresponding to only $m \leq k$, distinct eigenvalues of $AC^{-1}C^{-T}A^T$, then the method converges in at most m-iterations to a unique solution of $Ax = b$.

Proof

Assume that r_0 has non-zero components only along the eigenvectors $w_1, w_2, \ldots, w_k$ of $AC^{-1}C^{-T}A^T$, then

$$r_0 = \sum_{j=1}^{k} s_j w_j$$

Therefore

$$E(r_i) \leq P_i \, E(r_0), \quad P_i = \min_{R_i} \max_{1 \leq j \leq k} |R_i(\lambda_j)|$$

Therefore for $\lambda_1, \lambda_2, \ldots, \lambda_m$ distinct eigenvalues of $AC^{-1}C^{-T}A^T$ from the k-eigenvalues

$$P_i = \min_{R_i} \max_{1 \leq j \leq m} |R_i(\lambda_j)|$$

Therefore the polynomial $R_m(y) = \prod_{j=1}^{m} (\frac{y - \lambda_j}{\lambda_j})$ satisfy $R_m(\lambda_j) = 0$ which implies that $E(r_i) = 0$, for $i \leq m$.

A similar result we will have to theorem 5.3.4 as a result of applying the post-preconditioned conjugate gradient method as follows.

Theorem 5.4.4

At the i-th iteration we have

$$E(r_i) \leq 2(\frac{1 - \sqrt{p}}{1 + \sqrt{p}})^i E(r_0) \quad , \quad P = \frac{1}{k(C^{-T}A^TAC^{-1})}$$

5.4.2 Pre-preconditioned bidiagonalization technique

The second possible way to solve the system of linear equations 5.1.1 by using an approximate matrix factorization C for the matrix A, thus the linear least squares to solve 5.1.1 is

$$\text{minimize } ||C^{-1}Ax - C^{-1}b|| \qquad 5.4.2$$

where C is any approximate factorization for the n x n matrix A, or so find x and r such that $r + C^{-1}Ax = C^{-1}b$, $A^TC^{-T}r = 0$, any such x is called "least squares solution".

Where the minimum least square solution is the unique solution orthogonal to $N(C^{-1}A)$, and the iteration will be for x itself.

So if y is a solution of $A^TC^{-T}C^{-1}AA^TC^{-T}y = A^TC^{-T}b$, then $x = A^TC^{-T}y$ is the minimum least squares solution.

Suppose that

$$C^{-1}AV = UL \quad \text{and the representation } x = Vy \text{ holds,}$$

and since $r^TC^{-1}AV = r^T UL = 0$, where L is non singular,

Therefore $U^Tr = 0$

Hence $C^{-1}b = r + C^{-1}Ax = r + C^{-1}AVy = r+ULy$

Therefore x and y can be found from

$$Ly = U^Tb,\ x = Vy$$

Now to show that for a special choice of u_1, such a representation of x is possible, when the residual r is compatible (i.e. r is close to zero).

Therefore with respect to the equation $C^{-1}Ax = C^{-1}b$, $C^{-1}b \in R(C^{-1}A)$

solve $Cb^* = b$, for b^*, such that $b^* = C^{-1}b$

Let $u_1 = b^*/\beta_1$, $\beta_1 = ||b^*||_2$

Therefore $u_1 \in R(C^{-1}A)$, and $C^{-1}Ax = C^{-1}b = b^* = \beta_1 u_1$

$$A^TC^{-T}U = VL^T \quad \text{and} \quad C^{-1}AV = UL$$

where $UU^T = VV^T = I$

Let $x = Vy + w$; where $V^Tw = 0$

Then $C^{-1}Ax = ULy + C^{-1}Aw = \beta_1 u_1$

which implies that

$$Ly + U^TC^{-1}Aw = \beta_1 u_1$$

Hence $Ly + LV^Tw = \beta_1 u_1$

But since $V^Tw = 0$, therefore $Ly = \beta_1 e_1$, and this implies that

$C^{-1}Aw = 0$, and the solution has the form $x = Vy$

The elements of y are given by $Ly = \beta_1 e_1$ (i.e. $\eta_1 = \beta_1/\alpha_1$, and

$$\eta_{i+1} = -(\beta_{i+1}/\alpha_{i+1})\eta_i)$$

The second possibility to be examined now is the one when the residual r is non-zero (Incompatible).

Solve $Cb^* = b$

so that $u_1 = b^*/\beta_1$, where $u_1 \in R(C^{-1}A)$

Thus the problem is to solve, for x and r

$$r + C^{-1}A = C^{-1}b = \beta_1 u_1, \quad A^T C^{-T} r = 0$$

where $A^T C^{-T}\tilde{U} = V\tilde{L}$; $C^{-1}AV = \tilde{U}\tilde{L}$

$$\tilde{U} = [U_1, u_{k+1}]; \quad \tilde{L} = [L^T, \beta_{k+1} e_k]^T \quad \text{(see Paige (P4))}$$

such that $\tilde{U}^T\tilde{U} = I_{k+1}$, $V^TV = I_k$

Now assume that $x = Vy + w$; with $V^Tw = 0$

Therefore $r + \tilde{U}\tilde{L}y + C^{-1}Aw = \beta_1 u_1$, but $\tilde{U}^T C^{-1}Aw = \tilde{L}V^Tw$

So $\tilde{L}y = \beta_1 e_1 - \tilde{U}^T r$

Hence we conclude that

$$r + C^{-1}Aw = \tilde{U}\tilde{U}^T r$$

Thus $w^TA^TC^{-T}r + w^TA^TC^{-T}C^{-1}Aw = ||C^{-1}Aw||_2^2 = w^TA^TC^{-T}C^{-1}A\tilde{U}\tilde{U}^Tr = 0$

Thus $C^{-1}Aw = 0$, which implies that $x = Vy$

Algorithm 5.4.2

The pre-preconditioning bidiagonalization technique

1. $\tau_0 = 1$, $w_0 = 0$, $\eta_0 = -1$, $vz = 0$, $vw = 0$

   solve $Cb^* = b$, for b^*

   $\beta_1 u_1 = b^*$

   solve $C^T u_1^* = u_1$, for u_1^*

   $\alpha_1 v_1 = A^T u_1^*$

2. Iteration i

   $\eta_i = -\eta_{i-1}\beta_i/\alpha_i$ $\qquad vz = vz + \eta_i v_i$

   $w_i = (\tau_{i-1} - \beta_i w_{i-1})/\alpha_i$ $\qquad vw = vw + w_i v_i$

solve for $Cv_i^* = Av_i$, for v_i^*

$\beta_{i+1}u_{i+1} = v_i^* - \alpha_i u_i$

If $\beta_{i+1} = 0$, then (solution = vz, residual = 0, STOP)

$\tau_i = -\tau_{i-1}\,\alpha_i/\beta_{i+1}$

Solve for $C^T u_{i+1}^* = u_{i+1}$, for u_{i+1}

$\alpha_{i+1}v_{i+1} = A^T u_{i+1}^* - \beta_{i+1}v_i$

If $\alpha_{i+1} = 0$, then ($\gamma = \beta_{i+1}\eta_i/(\beta_{i+1}w_i - \tau_i)$), solution = $vz - \gamma vw$

3. Otherwise, set i = i + 1, go to 2

By following the same policy which has been used to prove the theorems 5.4.1, 5.4.2, 5.4.3 and 5.4.4, we can prove the following theorems.

Theorem 5.4.5

At the i-th iteration, we have the following

1(a) $A^T C^{-T} C^{-1} A v_i \in \{v_1, v_2, \ldots, v_{i+1}\}$

1(b) $C^{-1} A A^T C^{-T} u_i \in \{u_1, u_2, \ldots, u_{i+1}\}$

2 $A^T C^{-T} r_i \in \{v_1, v_2, \ldots, v_{i+1}\}$

3(a) $C^{-1} A v_i \in \{u_1, u_2, \ldots, u_{i+1}\}$

3(b) $A^T C^{-T} u_i \in \{v_1, v_2, \ldots, v_{i+1}\}$

4 $r_i \in \{u_1, u_2, \ldots, u_{i+1}\}$

5 $\{u_1, u_2, \ldots u_{i+1}\} = \{u_1, C^{-1}AA^TC^{-T}u_1, \ldots, (C^{-1}AA^TC^{-T})^{i+1}u_1\}$

$= \{r_0, C^{-1}AA^TC^{-T}u_1, \ldots, (C^{-1}AA^TC^{-T})^{i+1}r_0\}$

6 $\{v_1, v_2, \ldots, v_{i+1}\} = \{v_1, C^{-1}AA^TC^{-T}v_i, \ldots, (C^{-1}AA^TC^{-T})^{i+1}v_1\}$

Theorem 5.4.6

At the i-th step of iteration we have

$$E(r_i) \lesssim P_i E(r_0),\ P_i = I - C^{-1}AA^TC^{-T}P_{i-1}(C^{-1}AA^TC^{-T})$$

where $E(r_i)$ is the error in the residual at the i-th iteration.

Theorem 5.4.7

If r_0 has non-zero components along $k \lesssim m$ (for k-eigenvalues of $C^{-1}AA^TC^{-T}$) corresponding to only $m \lesssim k$, distinct eigenvalues of $C^{-1}AA^TC^{-T}$, then the method converges in at most m-iterations to a unique solution of $Ax = b$ (or $C^{-1}Ax = C^{-1}b$).

Theorem 5.4.8

At the i-th iteration, we have

$$E(r_i) \lesssim 2\left(\frac{1 - \sqrt{p}}{1 + \sqrt{p}}\right)^i E(r_0)\ ,\quad p = \frac{1}{k(A^TC^{-T}C^{-1}A)}$$

5.5 Numerical results and discussion

In this section we are going to study the numerical solution and the computational complexity of two non linear elliptic equations.

Problem 1 (see C10)

$$-u_{xx} - u_{yy} + (1 - e^{-5x})e^u = 1 \qquad 5.5.1$$

Equation 5.5.1 is to be solved on

$$R = \{(x,y) \mid 0 \lesssim x \lesssim 1,\ 0 \lesssim y \lesssim 1\}$$

with boundary conditions

$$u(x,y) = 0 \quad \text{on} \quad x = 0$$

$$u(x,y) = 1 \quad \text{on} \quad x = 1$$

$$u_y(x,y) = 0 \quad \text{on} \quad y = 0 \quad \text{and}$$

$$u(x,y) = \left\{ \begin{array}{ll} u_y(x,y) = 0 & \text{for } 0 < x \leq a < \frac{1}{2} \\ u(x,y) = -1 & \text{for } a < x < 1 - a \\ u_y(x,y) = 0 & \text{for } 1 - a \leq x < 1 \end{array} \right\} \text{ on } y = 1$$

In 5.5.1 the value of a was taken as a = 5/16

Problem 5.5.1 has been approximated by using five points finite difference approximation. At (i,j) 5.5.1 the approximation is

$$\frac{1}{h^2}(-u_{ij-1} - u_{i-1j} + 4u_{ij} - u_{i+1j} - u_{ij+1}) + (1 - e^{-5x_i}) \exp(u_{ij}) = 1 \qquad 5.5.2$$

(where h is the mesh size).

The system generated from the application of finite difference approximations is

$$Au = F(u) \qquad 5.5.3$$

where A is an n x n nonsymmetric matrix and F(u) is a non linear operator.

Problem 2

The second problem is the problem of determining the laminar flow of a viscous fluid in a square duct, the problem is (see Y2)

$$\frac{\partial}{\partial x}\left(w \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(w \frac{\partial u}{\partial y}\right) + \frac{f.R_e}{2} = 0$$

$$w = \left[(u_x)^2 + (u_y)^2\right]^{(n-1)/2} \qquad 5.5.4$$

$$\int_0^1 \int_0^1 u(x,y)\, dxdy = 1$$

defined on $R = \{(x,y) \mid 0 \leq x \leq 1,\ 0 \leq y \leq 1\}$

with boundary condition

$$u(x,y) = 0$$

The range of the value n in 5.5.4 is $0 < n \leq 1$, in the case n = 1 we will have a linear system.

We solved 5.5.4 with $f.R_e/2 = 1$ and because of the symmetricity of 5.5.4 (see Y2) it is sufficient to solve 5.5.4 in the square bounded by

$$0.5 \leq x \leq 1 \quad \text{and} \quad 0.5 \leq y \leq 1$$

The symmetricity of the region provides the additional boundary conditions

$$\frac{\partial u}{\partial x}\ (0.5,y) = \frac{\partial u}{\partial y}\ (x,0.5) = 0 \qquad 5.5.5$$

We used the same criteria of approximating 5.5.3 as in (Y2). The generated system from the application of five points finite difference to approximate (5.5.4) with (5.5.5) is as

$$A(u).u = C \qquad 5.5.6$$

where $A(u)$ is a matrix, each component is a function of the solution vector u, and C is a constant vector (when $n = 1$, we have $A.u = C$ where A is a matrix of constant coefficients).

The numerical methods used to solve problems 1 and 2 are

1. Non linearized S.I.P.
   (calculating the matrix factorization every iteration for problem 2)
2. Linearized S.I.P. for every 5-iterations
   (calculating the matrix factorization every 5 iterations for problem 2)
3. Linearized S.I.P. for every 10-iterations
   (calculating the matrix factorization every 10 iterations for problem 2)
4. Bidigonalization technique
5. Post-preconditioned bidiagonalization technique
6. Pre-preconditioned bidiagonalization technique

(Note: In the techniques 1, 2, 3,5 and 6 we used the original Stone factorization for symmetric and non symmetric factorization). The comparison between the above methods have been considered with regard to the following factors, just as before,

1. Number of arithmetics
2. Size of storage
3. Number of iterations
4. Time required

5.5.1 Number of arithmetics

As it has been described in chapter 4, the number of arithmetics is to be considered as

1. Information number of arithmetics
2. Linearization number of arithmetics
3. Algorithm number of arithmetics

Table 5.1 gives a complete list for the total number of arithmetics required by each algorithm.

5.5.2 Size of storage

This, we also again, consider with regard to

1. Information store units
2. Linearization store units
3. Algorithm store units
4. Convergence store unit
5. Outer iteration convergence unit

Table 5.2 gives a full description of the store units required for each algorithm.

Arithmetics	Non linearized S.I.P.	Linearized S.I.P. (at every 5 or 10 iter)	Bidiagonalization	Post preconditioned bidiagonalization	Pre preconditioned bidiagonalization
Information arithmetics	Prob 1-matrix factorization + product of L and U	Prob 1-matrix factorization + the product L and U (Prob 2 = 0)	-	Calculation matrix factorization	Calculating matrix factorization
Linearization arithmetics	-	Linearization (+ (prob 2) matrix factorization and product of L and U)	$(5N^2 - 2N) + 4N^2$	$4N^2 + (5N^2 - 2N)$ + forward and backward solver	$4N^2 + (5N^2 - 2N)$ + 2(forward and backward solver
Algorithm arithmetics	$12N^2 - 6N - 2$ + backward and forward solver (+ (prob 2) matrix factorization and product L and U)	$12N^2 - 6N - 2$ + forward and backward solver	$9N^2 + 2(5N^2 - 2N)$	$9N^2 + 2(5N^2 - 2N)$ + 2(forward and backward solver)	$9N^2 + 2(5N^2 - 2N) + 7$ + 2(forward and backward solver)
Convergence arithmetics	-	-	-	-	-

Table 5.1

The amount of arithmetics required to solve problems 1 and 2 using methods 1-6

Store Unit	Nonlinearized S.I.P.	Linearized S.I.P. (at 5 or 10 iterations)	Bidiagonaliztion	Post preconditioned bidiagonalization	Pre preconditioned bidiagonalization
Information store unit	12 vectors (prob 1) 0 vectors (prob 2)	Prob 1 - 12 vectors Prob 2 - 0	-	5 vectors	5 vectors
Linearization store unit	-	1 vector (prob 1) 12 vectors (prob 2)	2 vectors	3 vectors	4 vectors
Algorithm store units	4 vectors (+ 12 vectors (prob 2))	4 vectors	4 vectors	6 vectors	6 vectors
Convergence store units	-	1	-	-	-
Outer iteration convergence	-	-	-	-	-

Table 5.2

The store required by methods 1-6

5.5.3 Number of iterations and time required

Table 5.3 shows the variation in the time and the number of iterations required to solve the first problem by non linear S.I.P., linearized S.I.P. at every 5-iterations and linearized S.I.P. at every 10-iterations respectively at the optimal value of α. From table 5.3 we conclude that the linearized S.I.P. requires less time and more iterations than the nonlinearized S.I.P. and the linearized S.I.P. at every 5-iterations. The difference in the time was because of the evaluation of the non linear operator.

Problem 2 has been solved by using the same technique (see table 5.4). Table 5.4 shows the difference between the degree of nonlinearity (the n-value in 5.5.4) with respect to the time required and the number of iterations to solve a system of nonlinear equations of the same dimension but different nonlinearity power (i.e. n = 1.0 (linear), 0.7 and 0.5 (non linear)).

On the other hand table 5.4 shows the variation in the number of iterations and the time required to solve a certain system by using non linear S.I.P. technique, linearized S.I.P. at every 5-iteration and linearized S.I.P. at every 10-iteration, where the last technique required less iteration and less time than the first two techniques.

Figures 5.1, 2 and 3 shows the variation in the number of iterations and $\alpha(\alpha \in [0,1])$ to solve the first problem by methods 1, 2 and 3, also they show the location of the optimal value of α. Also figures 5.4, 5 and 6 shows the variation in the number of iterations and α, to solve the second problem by methods 1, 2 and 3. We solved the first problem by didiagonalization, post-preconditioned bidiagonalization technique and pre-preconditioned bidiagonalization technique, table 5.5 shows that, post-preconditioned bidiagonalization required less iteration than

the bidiagonalization and pre-preconditioned bidiagonalization technique, but more than using non linearized S.I.P. technique (figure 5.7 shows the variation in the number of iterations required to solve the first problem by different techniques). An example for the variation in the time and number of iteration required to solve problem 2 (see table 5.6) it shows that post preconditioned bidiagonalization technique required less time and iteration than using bidiagonalization and pre preconditioned bidiagonalization.

Figures 5.8 and 5.9 show the variation in the number of iterations and $\alpha(\alpha \in [0,1])$ (by using the original Stone's factorization for symmetric and non symmetric system) by the applications of post preconditioned and pre preconditioned bidiagonalization techniques respectively to solve problem 2. Therefore we can strongly prefer post preconditioned bidiagonalization technique to solve non symmetric linear or (non-symmetric Jacobian) non linear system of equations rather than pre-preconditioned bidiagonalization technique. But the post-preconditioned bidiagonalization technique requires more time than linearized S.I.P. at every 10 iterations.

Eventually the total arithmetics required by psot-preconditioned bidiagonalization is more than even the linearized S.I.P. at every 10 iterations.

Hence, the final conclusion, we would prefer linearized S.I.P. rather than post preconditioned bidiagonalization technique.

Matrix Dimension	Method 1		Method 2		Method 3	
	No of iter	Time	No of iter	Time	Not of iter	Time
72	11	0.165431	17	0.11896	18	0.10318
90	12	0.22529	15	0.12270	19	0.13270
110	14	0.31952	17	0.17814	20	0.16840
240	23	1.14388	20	0.42752	30	0.54103
420	34	2.94869	32	1.25147	34	0.63335

Table 5.3

The number of iterations and time required to solve problem 1 by methods 1, 2 and 3.

h^{-1}/n-value	Method 1		Method 2		Method 3	
	No of iter	Time	No of iter	Time	No of iter	Time
20/1.0	20	0.82409	20	0.35210	20	0.29290
20/0.7	29	2.90466	29	0.88154	30	0.62481
20/0.5	55	5.59646	50	1.50134	60	1.25810
40/1.0	69	1.86620	69	1.50464	69	1.32627
40/0.7	94	38.69199	99	11.579241	106	8.52697
40/0.5	149	61.50557	159	18.55301	876	69.75483

Table 5.4

The number of iterations and the time required to solve problem 2 by methods 1, 2 and 3

Matrix Dimension	Method 1	Method 2	Method 3	Method 4	Method 5	Method 6
72	11	15	18	372	42	90
90	12	15	19	468	36	102
40	14	17	20	686	39	121
240	23	20	30	1771	60	217
420	34	32	34	2982	54	307

Table 5.5 The final table to show the variation in the number of iterations to solve problem 1 by methods 1-6.

h^{-1}/n-value	Method 4		Method 5		Method 6	
	No of iter.	Time	No of iter.	Time	No of iter.	Time
20/1.0	147	0.99384	25	0.35479	31	0.37776
20/0.7	14389	96.19554	141	2.40188	181	2.61438
20/0.5	24912	162.17800	238	3.87708	286	3.96856
40/1.0	975	22.39710	45	2.09375	71	2.77129
40/0.7	*	*	217	12.3375	404	17.2286
40/0.5	*	*	400	21.7505	680	28.333

Table 5.6 The number of iterations and the time required to solve problem 2 by methods 4, 5 and 6

(* No convergence achieved after 300 seconds)

With reference to (Y2), problem 2 has been solved using A.D.I. method, with a fixed number of inner iterations and using the following convergence checking

$$\max \mid w - w^{(k)} \mid \leq 10^{-4} \qquad 5.5.7$$

for outer iteration.

To facilitate a comparison with the results of Young and Wheeler we used the following starting approximation vector

$$x^{0} = \sin \pi x \sin \pi y \qquad 5.5.8$$

and their convergence criterion for outer iteration (5.5.7). (See footnote)

Table 5.6 gives the number of iterations required by methods 1, 2 and 3 to satisfy 5.5.7. Also we included some results of Young and Wheeler.

h^{-1}/n-value	method 1	method 2	method 3	Young and Wheeler
20/1.0	8	8	18	8
20/0.7	20	19	20	36
20/0.5	26	28	28	64

Table 5.6 Number of iterations for problem 2 by methods 1, 2 and 3 and A.D.I.

The obvious observation on these results, that method 1 requires less iteration than A.D.I. method to solve the laminar flow problem. And overall no requirement for extra work in S.I.P. technique for an accelerated parameter as in A.D.I. method.

Note: For problem 1 the starting vector was that used by Concus et al in C1O.

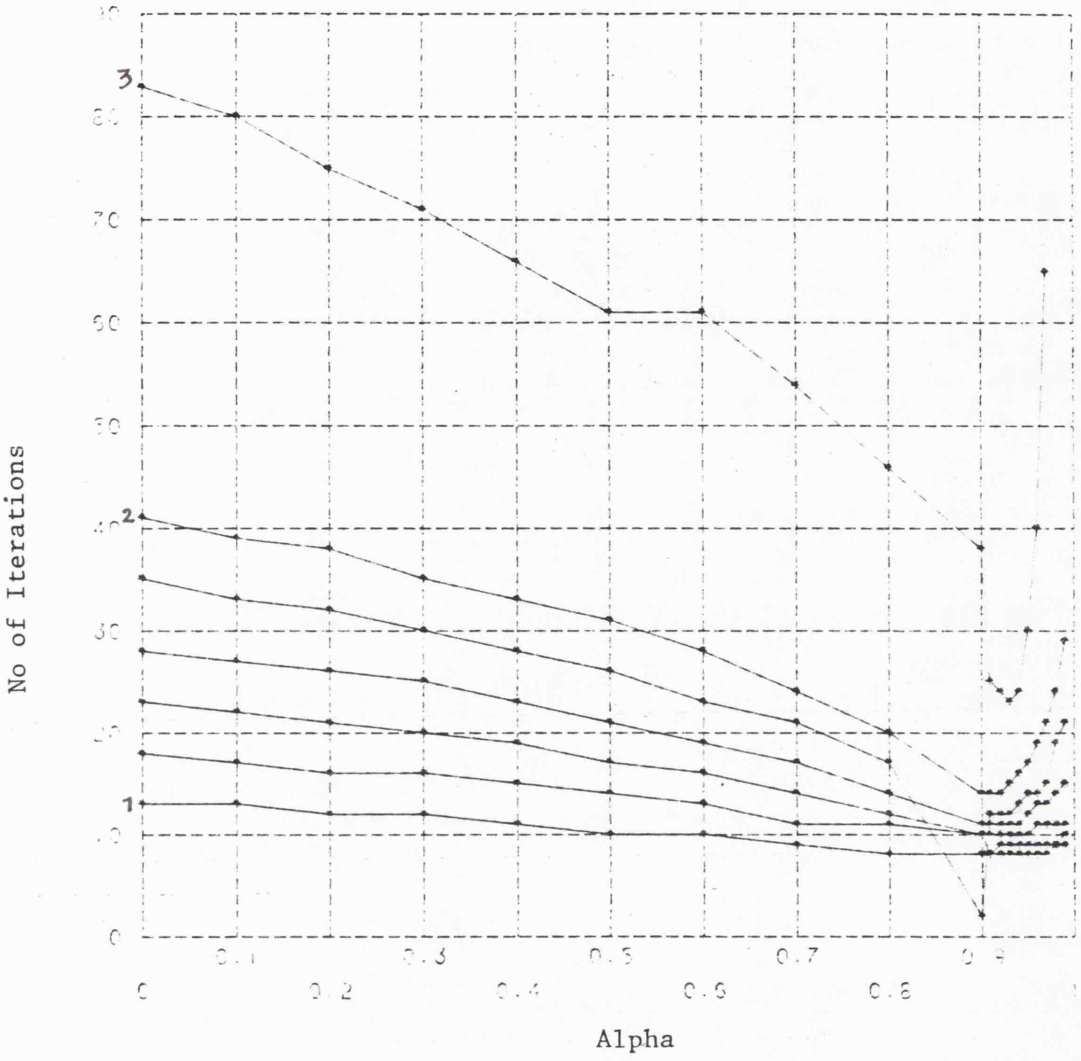


Fig. 5.1

The variation in the number of iterations with respect to $\alpha(\alpha \in [0,1])$ by using method 1 to solve the first problem

1. For a system of dimension = 20
2. For a system of dimension = 240
3. For a system of dimension = 420

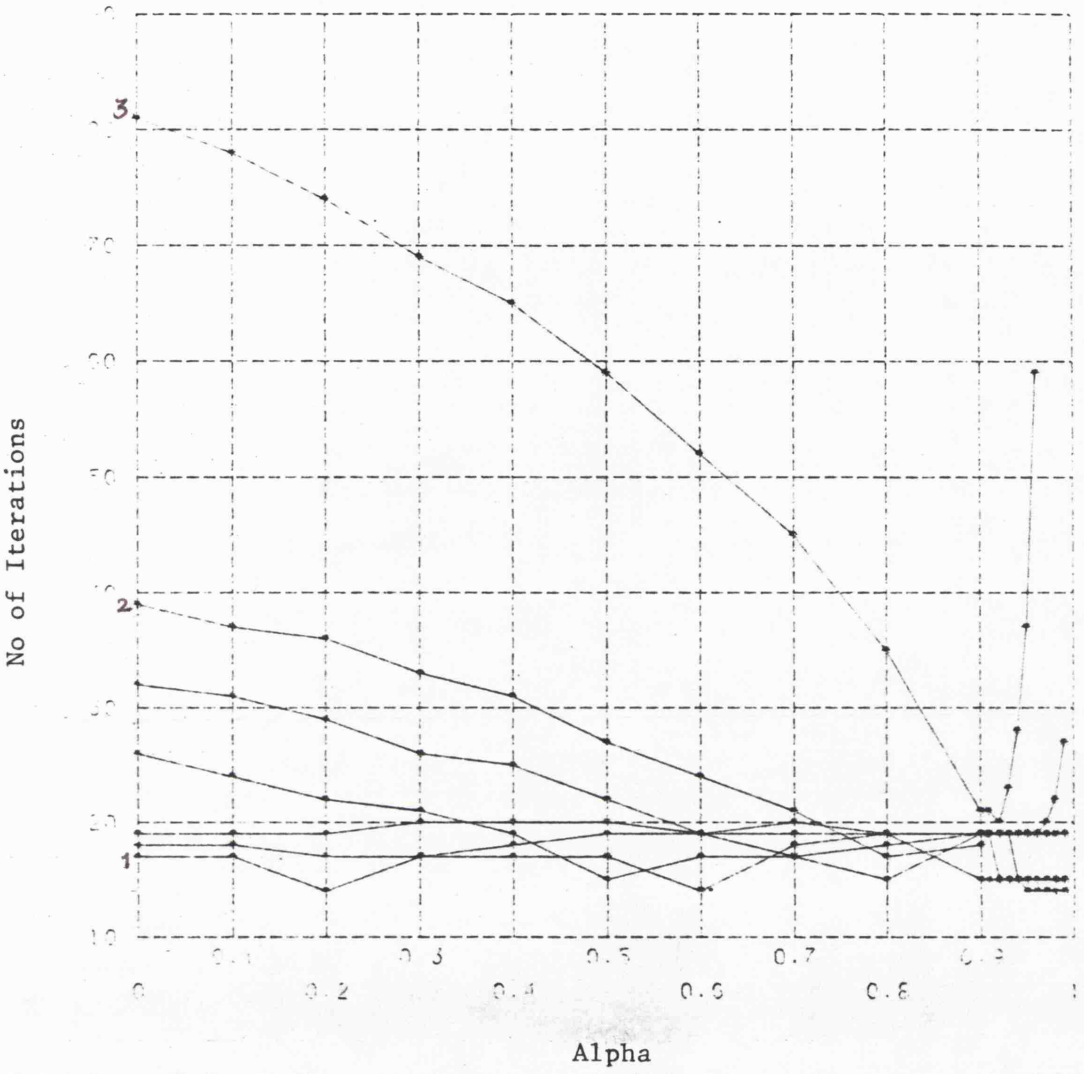


Fig. 5.2

The variations in the number of iterations with respect to $\alpha(\alpha \in [0,1])$ by using method 2 to solve the first problem

1. For a system of dimension = 20
2. For a system of dimension = 240
3. For a system of dimension = 420

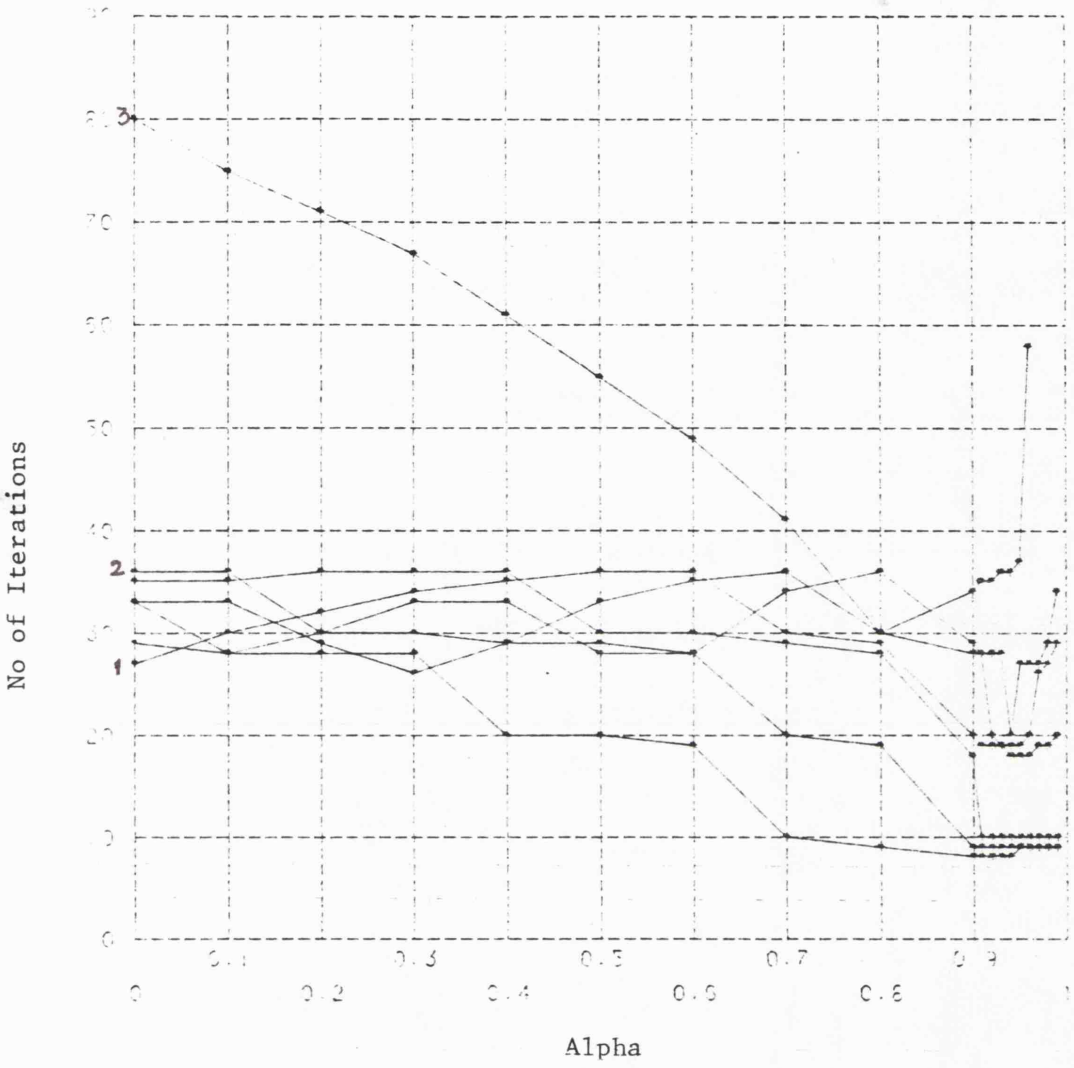


Fig. 5.3

The variations in the number of iterations with respect to $\alpha(\alpha \in [0,1])$ by using method 3 to solve the first problem

1. For a system of dimension = 20
2. For a system of dimension = 240
3. For a system of dimension = 420

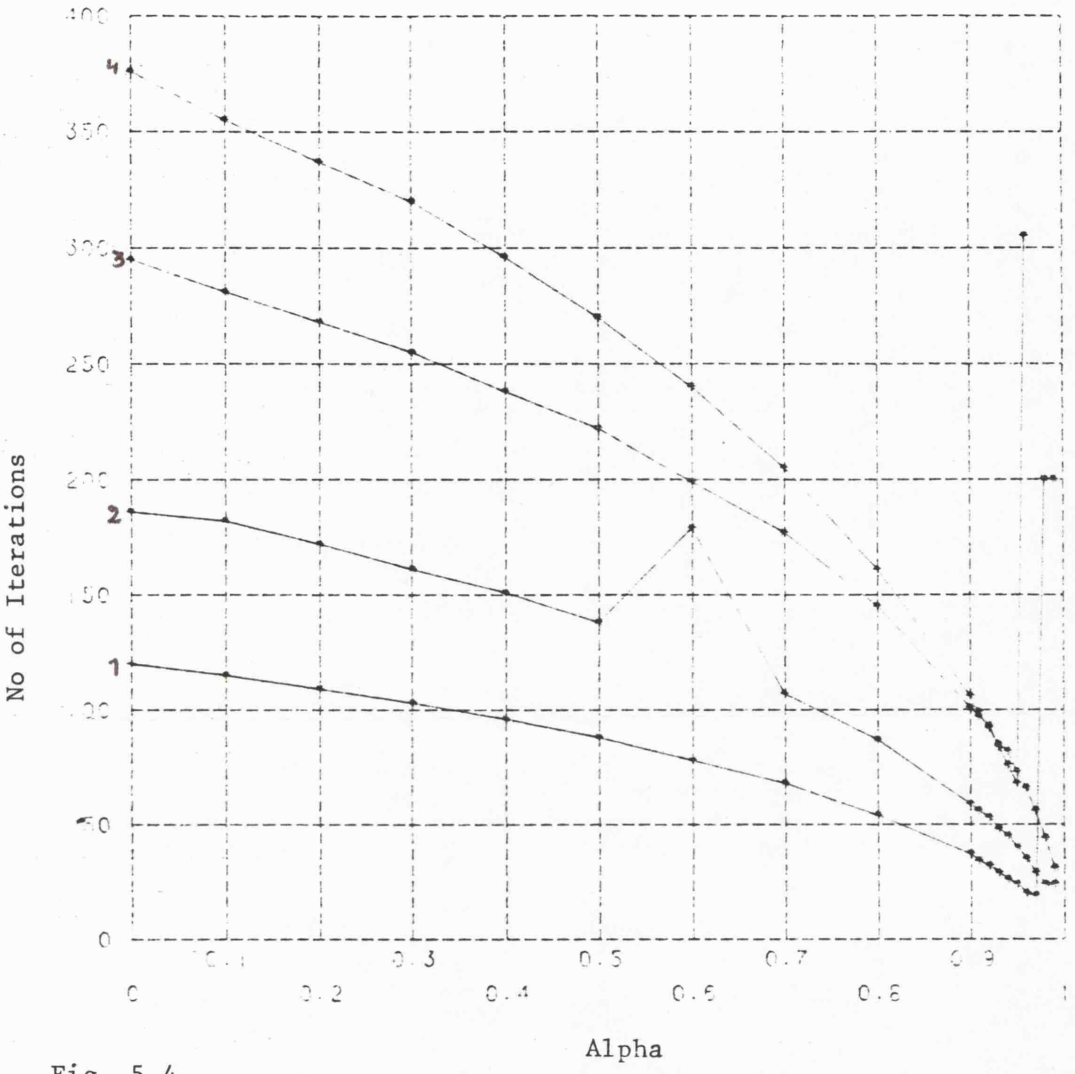


Fig. 5.4

The variation in the number of iterations with respect to $\alpha(\alpha \in [0,1])$ to solve problem 2 by method 1

1. For a system with $h^{-1} = 20$ and $n = 1.0$
2. For a system with $h^{-1} = 20$ and $n = 0.7$
3. For a system with $h^{-1} = 20$ and $n = 0.5$
4. For a system with $h^{-1} = 40$ and $n = 1.0$

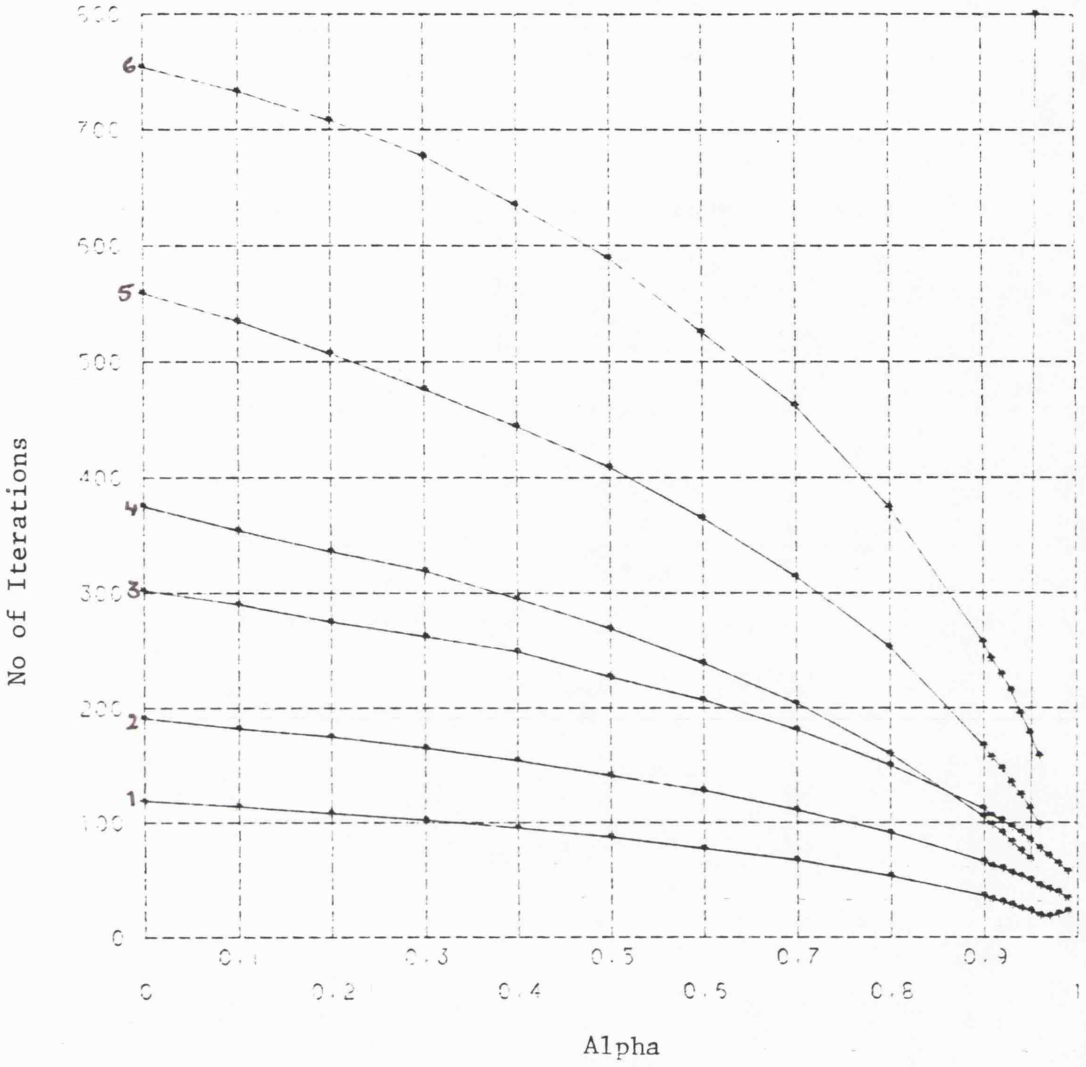


Fig. 5.5

The variation in the number of iterations with respect to $\alpha(\alpha \in [0,1])$ to solve problem 2 by method 2

1. For a system with $h^{-1} = 20$ and $n = 1.0$
2. For a system with $h^{-1} = 20$ and $n = 0.7$
3. For a system with $h^{-1} = 20$ and $n = 0.5$
4. For a system with $h^{-1} = 40$ and $n = 1.0$
5. For a system with $h^{-1} = 40$ and $n = 0.7$
6. For a system with $h^{-1} = 40$ and $n = 0.5$

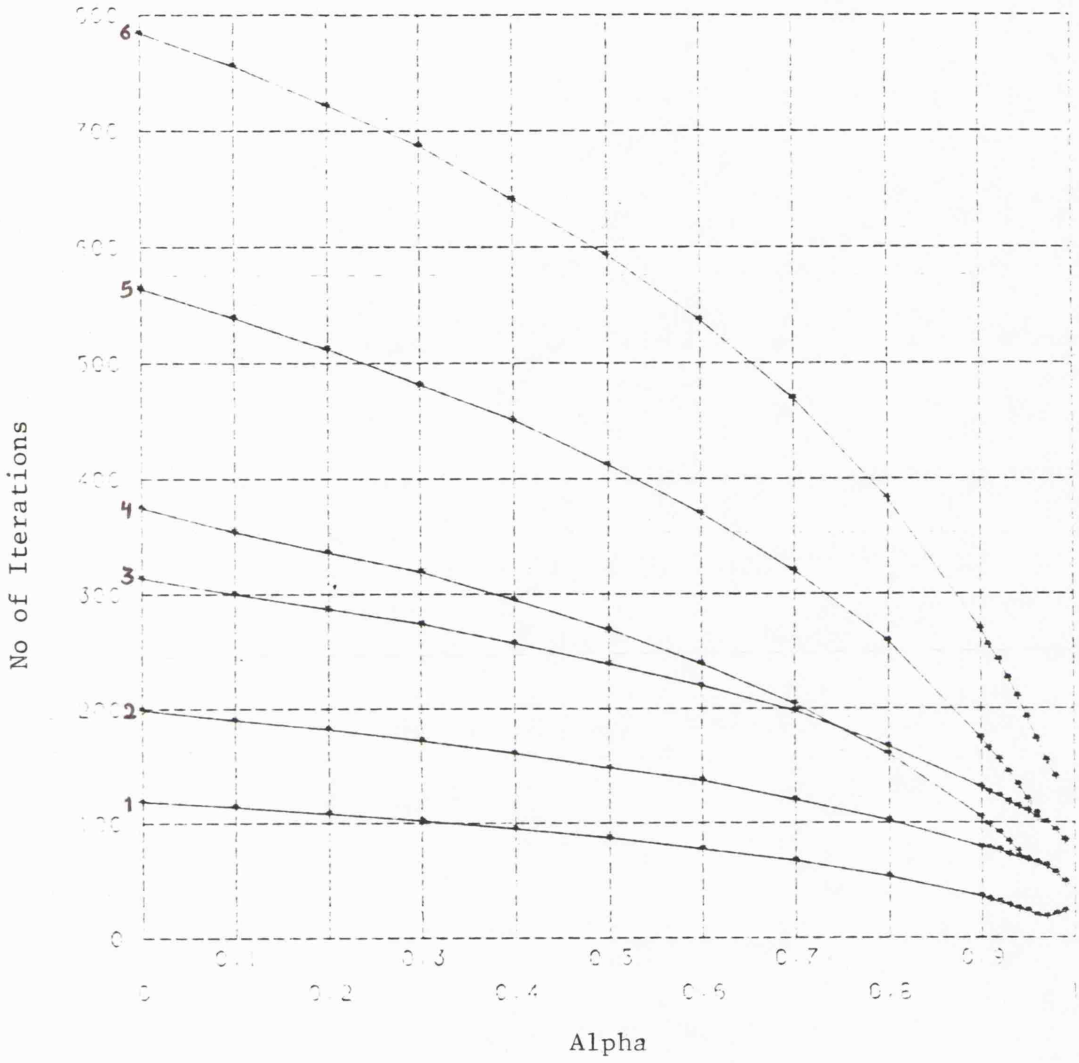


Fig. 5.6

The variation in the number of iterations with respect to $\alpha(\alpha \in [0,1])$ to solve problem 2 by method 3

1. For a system with $h^{-1} = 20$ and $n = 1.0$
2. For a system with $h^{-1} = 20$ and $n = 0.7$
3. For a system with $h^{-1} = 20$ and $n = 0.5$
4. For a system with $h^{-1} = 40$ and $n = 1.0$
5. For a system with $h^{-1} = 40$ and $n = 0.7$
6. For a system with $h^{-1} = 40$ and $n = 0.5$

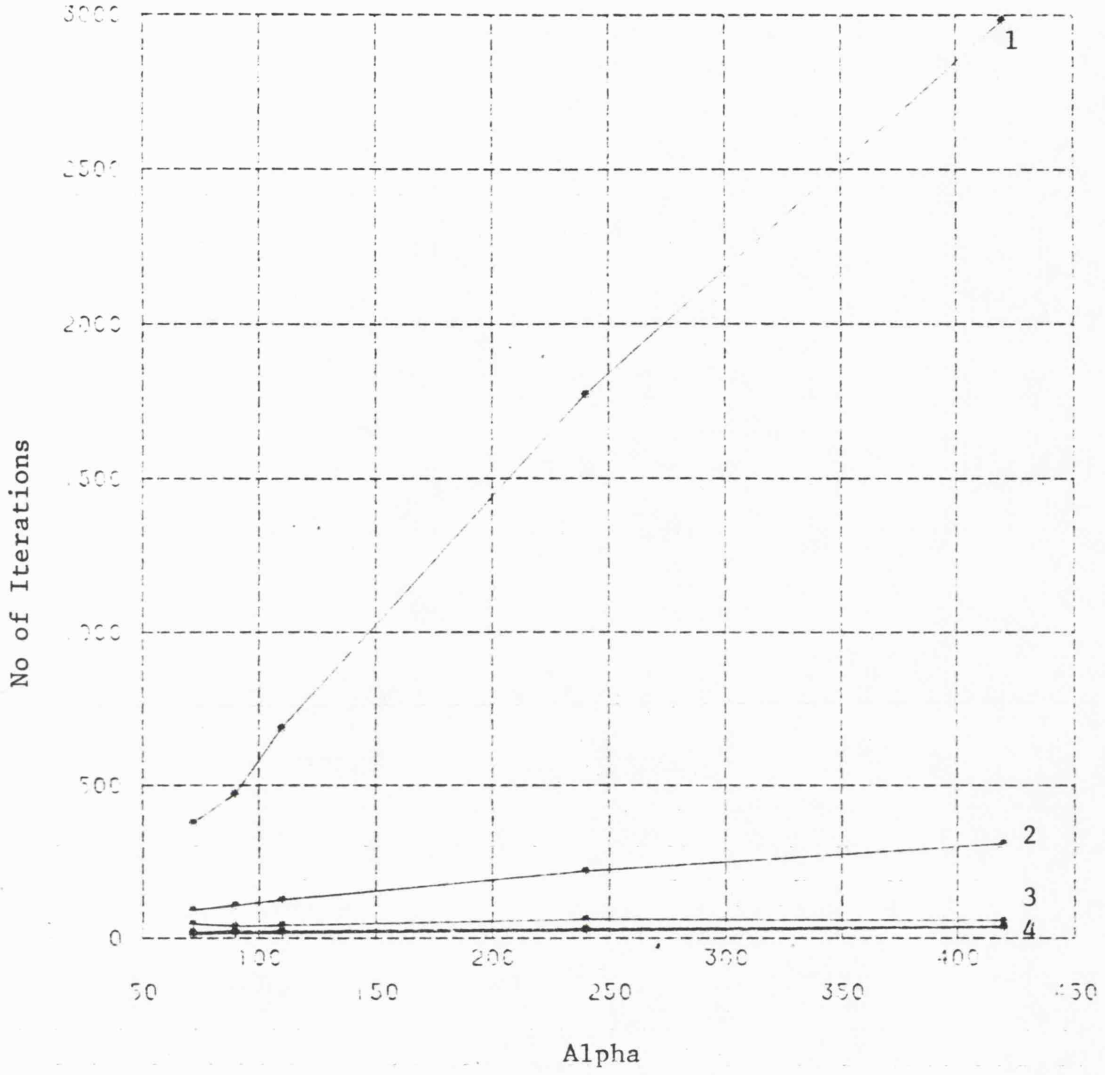


Fig. 5.7

The variation in number of iterations required to solve problem 1 by

1. Bidiagonalization technique
2. Pre-preconditioned bidiagonalization technique
3. Post-preconditioned bidiagonalization technique
4. S.I.P. (methods 1, 2 and 3)

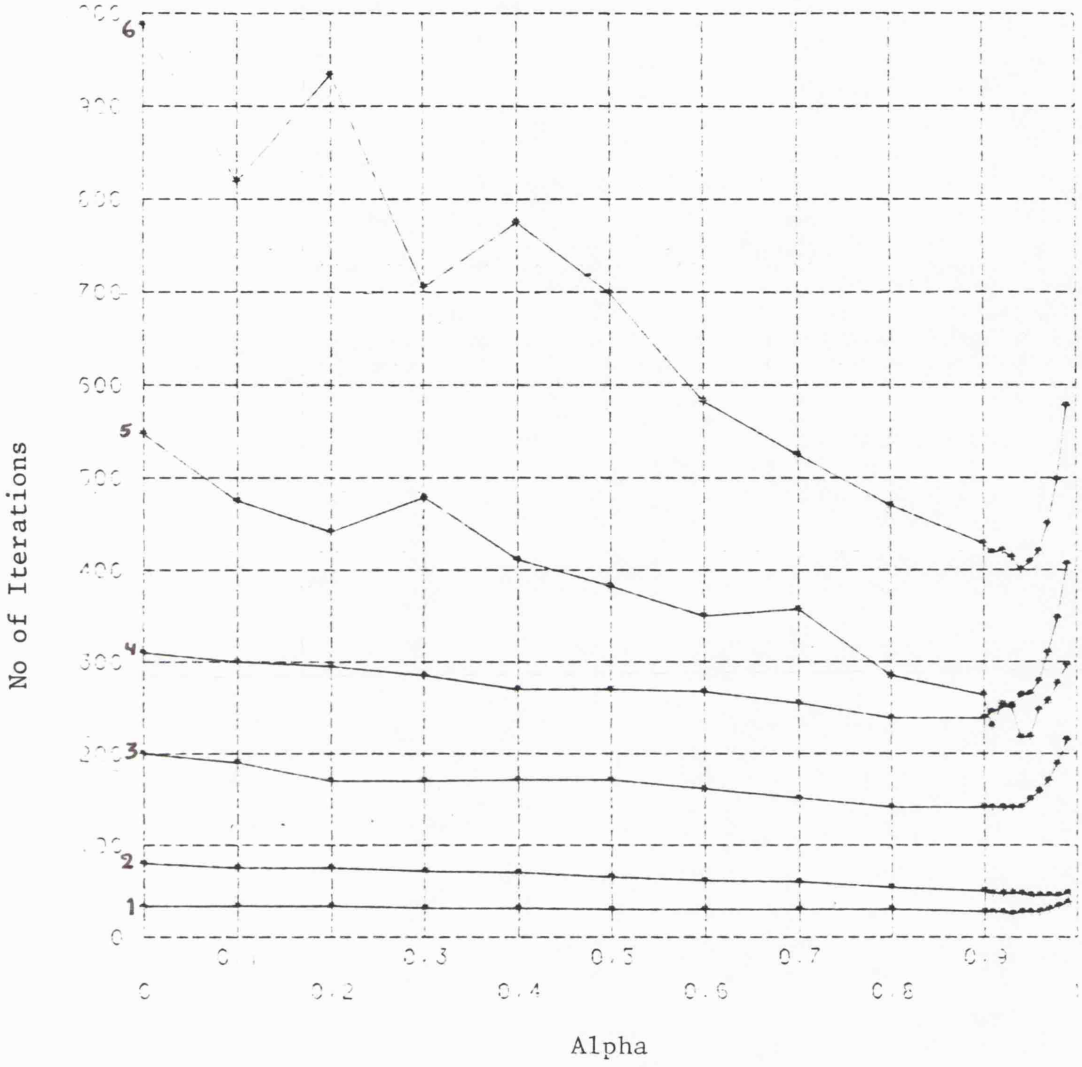


Fig. 5.8

The variation in the number of iterations with respect to $\alpha(\alpha \in [0,1])$ to solve problem 2 by post-preconditioned bidiagonalization technique

1. For a system with $h^{-1} = 20$ and $n = 1.0$
2. For a system with $h^{-1} = 20$ and $n = 0.7$
3. For a system with $h^{-1} = 20$ and $n = 0.5$
4. For a system with $h^{-1} = 40$ and $n = 1.0$
5. For a system with $h^{-1} = 40$ and $n = 0.7$
6. For a system with $h^{-1} = 40$ and $n = 0.5$

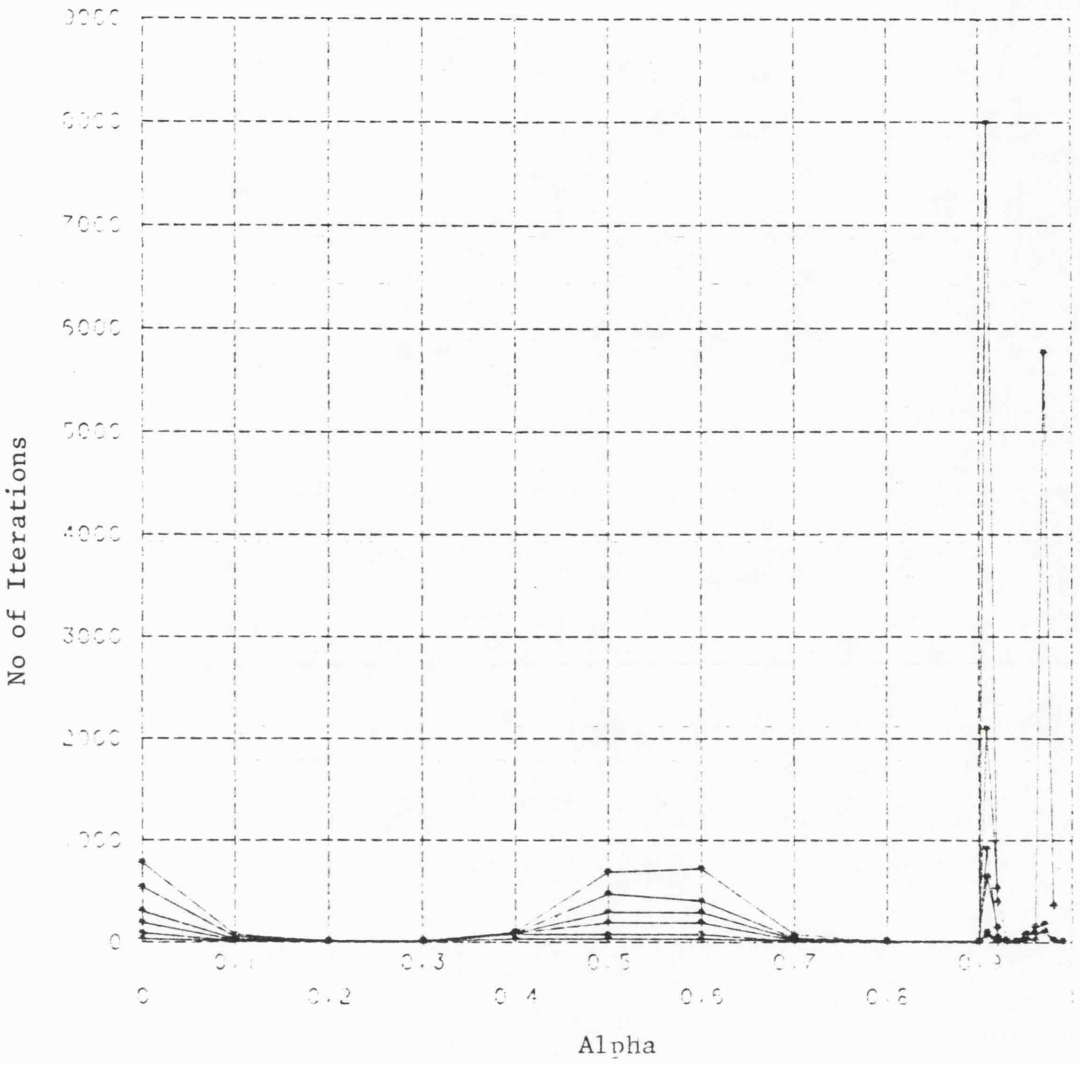


Fig. 5.9

The variation in the number of iterations with respect to $\alpha(\alpha \in [0,1])$ to solve problem 2 by pre-preconditioned bidiagonalization technique

CHAPTER 6 Multigrid Method

6.1 Introduction

In most numerical techniques for solving partial differential equations, the first step is to discretize the problem, choosing approximating algebraic equations on a finite dimensional approximation space, and then is devised a numerical process to solve (approximately) this system of discrete equations. The accuracy of the approximated solutions depends of course on the kind of the numerical technique.

In most numerical techniques there is no "real" interplay allowed between discretization and solution processes. This usually results in enormous waste. The discretization process being unable to predict the proper solution, and the order of approximation at each level being such as to produce a mesh which is too fine. Thus the algebraic system becomes unnecessarily large in size, while accuracy usually remains rather low.

On the other hand the solution process fails to take advantage of the fact that the algebraic system to be solved does not stand by itself but is actually an approximation to continuous equation. Thus, producing a numerical technique avoiding the above points is to study how to intermix discretization and solution process. Such a technique exists and is often called the "Multi Grid Method". This method is an iterative method that solves a system of discrete equations on a given grid by interactions between coarse and fine grids by taking advantage of the relation between different levels of discretized continuous problem.

The multigrid method can be viewed in two ways:

1) Consider the coarse grids as correction grids, accelerating the convergence of a relaxation scheme on the finest grid by liquidating smooth error components.

2) Regard finer grids as the correction grids improving accuracy on coarser grids.

The idea of using auxiliary systems of difference equations corresponding to coarser grids has been firstly described by Fedorenko (F1), in 1961. Since then this idea has only been revived by A. Brandt (B13), 1973, in what is called "Multi Level Adaptive Technique". He studied the application of multigrid method on discretized boundary value problems, obtained by 5-point formula, defined over different regions.

In 1976, W. Hackbush (H1) discussed the application of multigrid method for solving poisson equations defined on different regions. J.C. South and A. Brandt in 1976 (S12), studied the application of a multigrid method as a possible means of accelerating convergence in relaxation technique for a transonic flow.

Multigrid method has been applied to solve the system generated from the application of finite element method. For instance R.A. Nicolaides in 1975 (N4) studied the minimization of the positive quadratic form $a(u,u) - 2(u,f)$, by means of finite element methods, also he generalized the results in 1978 (N6) to cover the indefinite case.

W. Hackbush (H2) studied the application of multigrid method to a general linear elliptic equation which is subject to arbitrary boundary conditions defined on a rectangular region and also demonstrated that non-linear boundary value problem can be treated by a combination of multigrid iteration and Newton's method.

Two excellent papers appeared by A. Brandt in 1977 (B14) and (B15). In (B14) Brandt studied the solution of the boundary value problems, a full description of the multi-level adaptive technique has been given

together with the application to solve a non-linear equation. In (B15) Brandt gave an excellent survey of solution techniques which simultaneously use a sequence of increasing finer discretizations (finite element or finite difference) for the same continuous problem, also a flexible and economic multigrid data structure is presented together with the software developed to support grid manipulations in general, and multi-level adaptive technique in particular.

W. Hackbush in 1978 (H5) studied the multigrid method of second kind as a fast numerical algorithm for solving problems that can be represented to second kind elliptic boundary value problems formally as second kind freedholm integral equations, in 1979, Hackbush (H4) represented the application of multigrid method to solve an elliptic control problem with a quadratic cost function which require the solution a system of two elliptic boundary value problems.

In (H3) Hackbush applied multigrid method to solve Helmholtz's equation in a general region and to a differential equation with variable coefficients subject to arbitrary boundary conditions. In 1979 Hackbush (H6) demonstrated the application of multigrid method of second kind to elliptic boundary value problems and studied the treatment of non-linear boundary value problems.

In the same year another work appeared by Brandt (B16), was mainly concentrated the application of multi-level adaptive technique to solve a singular perturbation problem, and in the same year Brandt (B17) surveyed the state of art emphasizing steady state fluid dynamics applications, from slow viscous flows to transonic ones and various new techniques have been discussed including multi-level adaptive technique.

Wesseling in 1979 (W3) studied the rate of convergence and the computational complexity of multigrid method, a full description and explanation has been given by R.A. Nicolaides (N10) of a simple multigrid algorithm for solving finite element system.

In 1979 another excellent paper appeared by A. Brandt and N. Dinar (B19) wherein they studied and reviewed various concepts of ellipticity of finite difference approximations to general elliptic partial systems
rules are given, as well, for the construction of stable scheme with high approximation orders, even for singular perturbations problems, fast multigrid solvers for these discrete schemes are described.

A sequel to Brandt (B19) in (B18) Brandt defined and discussed the numerical stability of difference approximations to general boundary value problem regular as well as singular perturbation and non-elliptic problems. He proved that stability to be necessary and sufficient for fast multigrid solutions.

In 1980 Hackbush (H7) presented criteria of convergence that apply to general difference schemes for boundary value problems in Lipschitzian regions and convergence is proved for the multigrid algorithm with Gauss-Seidel's as smoothing procedure.

Hemker (H10) in 1980 considered the combination of incomplete LU-decomposition as a relaxation method in multigrid technique. Fuchs (F6) solved two transonic model equations by a combined multigrid method with Newton's method as a relaxation technique.

6.2 Multigrid method

The multigrid method is a systematic method of mixing relaxation sweeps with approximate solution of residual equations on coarser grids. The residual are in turn also solved by combining relaxation sweeps with

corrections through still coarser grids etc.

Suppose that the differential problem of the form

$$Au = F \text{ in } \Omega$$

$$Bu = C \text{ in } \partial\Omega \qquad 6.2.1$$

and this problem has been approximated by difference equations of the form

$$A^k u^k = F^k \quad \text{for } x \in \Omega^k$$

and $$B^k u^k = C^k \quad \text{for } x \in \partial\Omega^k \qquad 6.2.2$$

where Ω^0, $\Omega^1,\ldots,\Omega^M$ are the sequence of approximating the domain Ω with corresponding mesh sizes $h_0 > h_1 > \ldots > h_M$ such that $h_{k+1}:h_k = 1:2$.

Our target is to solve the discretized problem on the finest grid Ω^M, the main idea of multigrid method is to exploit the fact that the discrete problem on a coarser grid Ω^k, for example, approximates the same differential problem and hence can be used as a certain approximation to the Ω^M problem, the application of this fact is by solving the problem on Ω^k, and then interpolate the solution from Ω^k to Ω^M, a more advanced technique was to use a still coarser grid in a similar manner when solving the Ω^k problem, and so on (see (B14)).

Also, in the multigrid method, the role of relaxation is not to reduce error but to smooth it out i.e. to reduce the high frequency components of the error for more about smoothing the error (see (B14).

The main point in multigrid method is the "cycling" operations between the levels, or interpolation, for example from level k to k' will generally be denoted by $I_k^{k'}$. That is, if u^k is a function defined on the grid with mesh size h_k then $I_k^{k'} u^k$ is an approximation to u^k defined on the grid with mesh size $h_{k'}$. In particular I_k^{k+1} will denote an

interpolation polynomial from Ω_k into Ω_{k+1}, defined as (see H3)

$$I_k^{k+1}: \Omega_k \longrightarrow \Omega_{k+1}$$

$$(I_k^{k+1} u)_{i,j} = \sum_{\ell,k=-1}^{+1} \phi_{ij}^{\ell k} \, u_{i+\ell, j+k} \qquad 6.2.3$$

where $\phi_{ij}^{00} = 1$, $\phi_{ij}^{\pm 1,0}$ $(\phi_{ij}^{0,\pm 1}) = \frac{1}{2}$, $\phi_{ij}^{\pm 1,\pm 1} = \frac{1}{4}$, in the regular region.

But if $((i + 2)h, jh)$ or $(ih, (j + 2)h)$ ϵ Ω, then $\phi_{ij} = 0$

And in the irregular region if $((i + s)h, jh)$ ϵ Ω, then $\phi_{ij}^{1,0} = (s - 1)/s$.

The fine to coarse transfers I_k^{k-1} is made by some local averaging i.e. $I_k^{k-1} u(x^k)$ is some weighted average of values v^k at several points x^{k-1} close to x^k.

In the technique, the discretized system on coarser grids will be modified by changing their right hand side where in the correction scheme u^k is designed to be an approximate correction to u^{k+1}, hence the modified right hand side will be

$$f^k = I_{k+1}^k r^{k+1} \qquad 6.2.4$$

where $r^{k+1} = F^{k+1} - A^k u^k$ is the residual function of the current approximation u^{k+1} at the finer level.

The difference

$$\delta^k = f^k - F^k \qquad 6.2.5$$

gives an estimate for the local truncation error on level k, i.e. it is an approximation to

$$\delta_0^k = F^k - A^k u \qquad 6.2.6$$

The multigrid method unlike cycling algorithms, work themselves up from the coarsest level 1 to the finest level M, at each stage we will denote by ℓ the "currently finest" level.

1. Set $\ell = 1$, compute an approximate solution u^1 to the coarse grid equations, by relaxation or by some direct method (i.e. non iterative methods). If the system is non-linear the direct method means a few Newton iterations, where the linear system at each iteration is solved directly.

2. Increase ℓ by 1 (if $\ell = M$, the algorithm is terminated). Introduce, as the first approximations for the new finest level, the interpolation

$$u^{\ell} = I^{\ell}_{\ell-1} u^{\ell-1} \qquad 6.2.7$$

Then improve u^{ℓ} by one relaxation sweep, if the convergence at the current operation level has been achieved, then

$$u^{\ell}_{NEW} = u^{\ell}_{OLD} + I^{\ell}_{\ell-1} (u^{\ell-1} - I^{\ell}_{\ell-1} u^{\ell-1}_{OLD}) \qquad 6.2.8$$

and then start the process again

Otherwise we should do some more relaxation sweep, but in the case when the convergence rate is slow and $\ell > 1$, then decrease ℓ by 1. Introduce, as the first approximation for the new (coarser) level the iterative

$$u^{\ell} = I^{\ell}_{\ell+1} u^{\ell+1} \qquad 6.2.9$$

and then go to step 2 and start the same process.

The above technique was the main "Multi Grid" technique used by Brandt (for example, see (B19)).

Now we are going to consider the "Multi Grid" technique proposed by Hackbush (see H3).

Algorithm 6.2.1

Assume that the multigrid iteration is defined on the level k - 1 and let $u_k^i \in \Omega^k$ be a starting vector.

Iteration i + 1

$$u_k^{i+\frac{1}{2}} = A_k(u_k^i) + f_k$$

$$R_k = u_k^{i+\frac{1}{2}} - A_k(u_k^{i+\frac{1}{2}}) - f_k \in \Omega^k$$

$$R_{k-1} = I_k^{k-1} R_k \quad \text{(R is the current residual)}$$

Let $w_{k-1} \in \Omega^k$, be the solution of

$$w_{k-1} = A_{k-1}(w_{k-1}) + R_{k-1}$$

We approximate w_{k-1} by two iterations of multi grid algorithm on the level k - 1, starting with $w_{k-1}^{(0)} = u_{k-1}$.

Denote the result by $w_{k-1}^{(2)}$, we obtain the next iterate of u_k by

$$u_k^{i+1} = u_k^{i+\frac{1}{2}} - I_k^{k-1}(w_{k-1}^{(2)} - u_{k-1})$$

A comprehensive alteration has been introduced by Brandt (B19) to produce what he called "Full Multi Grid" method to solve any system of equations (linear or non-linear) such that, prespecified tolerance is introduced to switch to the coarser level ℓ - 1 with respect to the "Current Residual".

Also he proposed another criteria to switch to the coarser level ℓ - 1, when a pre-assigned number of relaxation sweeps on level ℓ has been completed, similarly the switch to finer level ℓ + 1 may be made as soon as a pre-assigned number of relaxation sweeps has been made on the level ℓ since the last visit to the finer level.

6.3 Multigrid method with S.I.P. relaxation technique

We consider the S.I.P. method as a relaxation method for one with multi grid algorithm to solve a sparse linear system (problem 2, at n = 1.0 , see chapter 5) and a sparse (Jacobian of) non-linear system (problem 2 at n = 0.7 and 0.5, see chapter 5).

The multigrid algorithm with S.I.P. method at level k to solve problem 2, where the generated system is A(u)u = C.

Algorithm 6.3.1

Assume that the multigrid iteration is defined on level k - 1, and let $u_k^i \in \Omega^k$ be a starting vector.

1. Iteration i +1

$$(A + B)_i \; u_k^{i+\frac{1}{2}} = (A + B)_i \; u_k^i - (A(u_k^i)\; u_k^i - C_k)$$

$$R_{k-1} = I_k^{k-1} R_k$$

Let $w_{k-1} \in \Omega^k$ be the solution of

$$A(w_{k-1})\; w_{k-1} = C_{k-1} + R_{k-1}$$

We approximate w_{k-1} by two (or three) iterations of multigrid algorithm on the level k - 1, starting with $w_{k-1}^{(0)} = u_{k-1}$.

The next iterate of u_k is

$$u_k^{i+1} = u_k^{i+\frac{1}{2}} - I_k^{k-1} \; (w_{k-1}^{(2)} - u_{k-1})$$

2. If the convergence achieved, STOP, otherwise set i = i + 1, and go to 1.

(Note: In algorithm 6.3.1 the matrix $(A + B)_i$ is the product of the Stone's factorization(s) L_i and U_i of the matrix $A(u_i)$.

Also we can solve the mildly non-linear equations (see chapter 4) by using multigrid algorithm with S.I.P. technique by using A. Bracha-Barak and P. Saylor (B10) factorization of symmetric matrix.

But in this work we preferred to consider a nonsymmetric matrix rather than symmetric for a general conclusion from the application of multi grid with S.I.P. as a relaxation procedure.

6.4 Numerical results and discussion

We consider the numerical solution of problems 2 (see chapter 5) by using multi grid algorithm with the following relaxation technique

1) Non-linearize S.I.P. technique
2) Linearized S.I.P. at every 5-iteration
3) Linearized S.I.P. at every 10-iteration

The multi grid algorithm has been performed for 3 levels i.e. M = 3 and the solution vector at each coarser grid has been produced with a full accuracy.

6.4.1 Number of arithmetics

With respect to the total number of arithmetics, because we are using the same smoothing technique, this can be viewed through the total number of iterations (including the number of iterations at each coarser grid) to solve the system at level M, plus the arithmetics required to perform the prolongation and restriction operations (the prolongation and restriction required $5N_{\ell}^2 - 2N_{\ell}$ - multiplications, where ℓ is the index of the finer level).

6.4.2 Storage units

Using multi grid method, to solve linear or non-linear system of equations, with any of the above relaxation techniques, require more store than using the relaxation technique as an iterative method to solve a system of linear (non-linear) equations.

6.4.3 Number of iterations and time required

We have considered the number of iteration and time required to solve a system of linear equations (problem 2, n = 1.0) and a system of non-linear equations (problem 2, n = 0.7 and 0.5).

The results from the numerical application of multigrid method to solve problem 2 are presented in Table 6.1, by using non linearized S.I.P. (method 1), linearized S.I.P. at every 5 iteration (method 2) and linearized S.I.P. at every 10 iteration (method 3) as relaxation techniques and comparatively with the result from the numerical application of method 1, method 2 and method 3 as an iterative technique, it shows that by using multigrid method to solve a linear system (n = 1.0) the reduction in the number of iteration was about 70% but the reduction in the time wasn't that impressive. But from solving the non-linear system when n = 0.7, the reduction in the number of iteration was 50% at least, and with regard to the time, using method 1 (i.e. non linearized S.I.P.) provides saving of 23%.

As the degree of nonlinearity increases (i.e. $n \longrightarrow 0$, with respect to problem 2, the reduction in time and the reduction in the number of iterations increases (with respect to each relaxation technique).

At n = 0.5 for instance the result shows average 75-94% reduction in the number of iteration and 50-86% reduction in the time.

Figures 6.1 and 6.2 show the variation in the number of iterations, to solve problem 2, by using multigrid method (with methods 1, 2 and 3 as a relaxation technique) and methods 1, 2 and 3 as an iterative technique respectively.

Eventually in addition to the total number of iterations required to solve any given problem, by multigrid method there is an implicit iteration to be performed, at the auxiliary levels, which should also be taken into consideration.

But the final judgement on the efficiency of a method concerns the size of storage required, the number of iterations required, and the total time taken.

Hence from the above comparison multigrid technique is found to be "cheap" to use, but difficult to programme, especially if we have an irregular region.

h^{-1}/n-value	MG + Method 1		MG + Method 2		MG + Method 3	
	No of iter.	Time	No of iter.	Time	No of iter.	Time
	(26)*		(26)*		(26)*	
40/1.0	48	10.31689	48	4.49583	48	3.66057
	(60)*		(47)*		(65)*	
40/0.7	35	29.64655	59	11.01685	54	7.71167
	(35)*		(72)*		(100)*	
40/0.5	53	28.93850	38	9.34351	52	9.48560

Table 6.1

The number of iterations and time required to solve problem 2 by multigrid method (with methods 1, 2 and 3 as a relaxation technique).

* (the total number of iterations at the coarser levels)

h^{-1}/n-value	Method 1		Method 2		Method 3	
	No of iter.	Time	No of iter.	Time	No of iter.	Time
40/1.0	69	11.4662	69	4.60464	69	3.72627
40/0.7	94	38.69199	99	11.57924	106	8.52697
40/0.5	149	61.50597	159	18.55301	876	69.95485

Table 6.2

The number of iterations and time required to solve problem 2 by the iterative methods 1, 2 and 3.

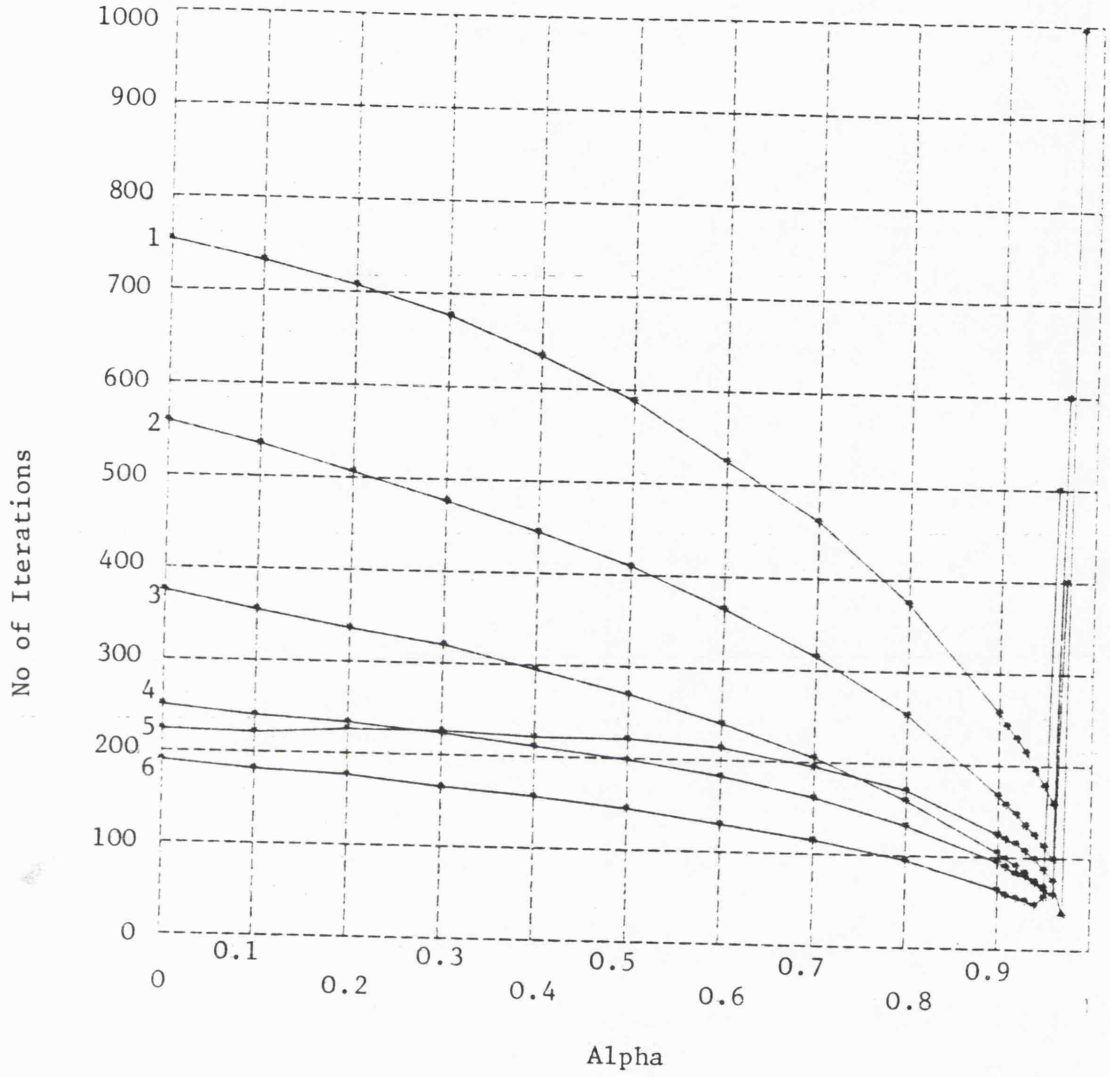


Fig. 6.1

The difference in number of iterations to solve laminar flow problem by multigrid method

1,4-40/0.5 by MG + method 2 and method 2
2,5-40/0.7 by MG + method 2 and method 2
3,6-40/1.0 by MG + method 2 and method 2

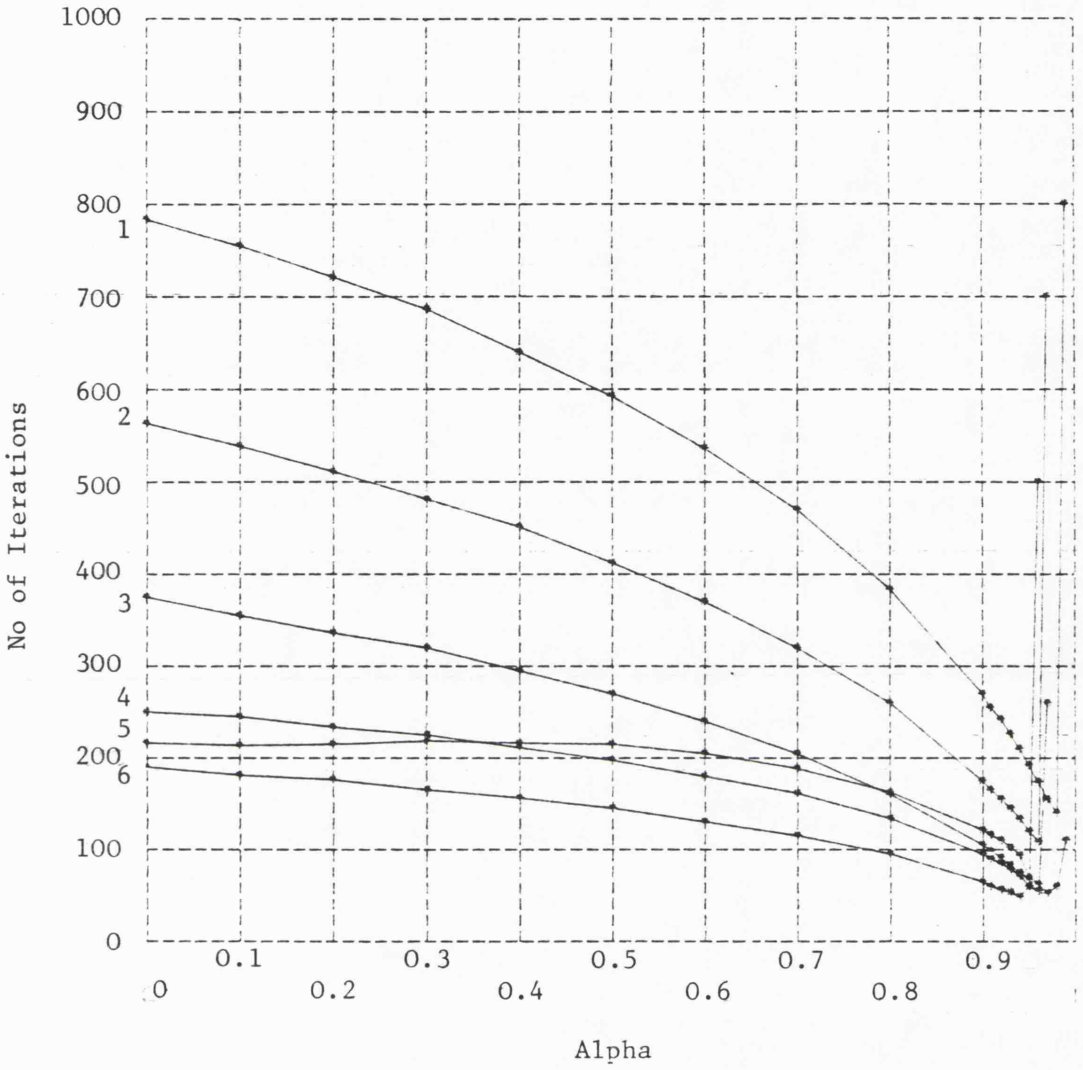


Fig. 6.2

The difference in number of iterations to solve laminar flow problem by multigrid method

1,4-40/0.5 by MG + method 3 and method 3
2,5-40/0.7 by MG + method 3 and method 3
3,6-40/1.0 by MG + method 3 and method 3

CHAPTER 7 Conclusions and Suggestions for Future Work

In this thesis we have used several numerical techniques to solve a system of non-linear equations, generated from the discretization of non-linear elliptic partial differential equations using finite difference approximations.

The iterative techniques which have been used to solve the systems are as follows:

1. Linearization Techniques
2. Non-linearization Technique

The numerical results contained in this thesis may be summarized as follows:

1. S.I.P. technique which has been proposed by H.L. Stone (S15) to solve a system of linear equations (using Stone approximate matrix factorization or its development by Bracha-Barak and Saylor (B10)). We succeeded in generalising this scheme to solve a system of non-linear equations, provided the starting iterative vector is chosen within an interval bounded by the minimum and the maximum values of the boundary condition(s), to what we called "non-linearized S.I.P.".

2. A generalization to the preconditioned linear conjugate gradient method has been proposed by P. Concus et al (C10), to what he called the "Preconditioned Non Linear Conjugate Gradient Method". We present the application of this technique using the improved Stone approximate matrix factorization (B10), to solve systems of nonlinear equations with symmetric and positive definite Jacobian.

3. A comparison between several iterative techniques (e.g. Non Linearized S.I.P., Linearized S.I.P., Linear Conjugate Gradient Method, Non Linear Conjugate Gradient Method and Preconditioned Non Linear Conjugate Gradient Method) has been studied with regard to the solution of mildly non linear elliptic equations.

The number of arithmetics, storage units, total number of iterations and time required to achieve the convergence are factors of comparison between the above iterative techniques.

We solved three models of mildly non linear elliptic equations (which have been earlier solved by Hageman and Porsching (H8), using non linear point (block) Gauss Seidel and non linear point (block) S.O.R. iterative techniques), the numerical results show that the most efficient iterative technique is non linearized S.I.P. technique.

Also compared with the numerical results of (H8), the nonlinearized S.I.P. method requires less iterations than nonlinear point (block) Gauss Seidel and non linear point (block) S.O.R. methods.

4. Non symmetricity is one of the problems in solving linear or non linear system. Paige proposed the "Bidiagonalization Technique" to solve a linear system $Ax = b$ (where A is $m \times n$ - matrix, $m \geq n$) (see P1). Numerical experiments on non linear and linear systems of equations on two model problems (problem 1 (see C10) , problem 2 (see Y2)) show the inefficiency of the bidiagonalization technique (see Chapter 5) with respect to the total number of iterations and time required to achieve the convergence.

We suggested two preconditioning techniques, to accelerate the convergence of the bidiagonalization technique, using Stone's approximate matrix

factorization (S15), these preconditioning are:

1) Post preconditioned bidiagonalization technique

2) Pre preconditioned bidiagonalization technique

We solved the two model problems (C10 and Y2) using nonlinearized S.I.P., bidiagonalization, post preconditioned bidiagonalization and pre preconditioned bidiagonalization techniques.

From the numerical results, post preconditioned bidiagonalization technique required less time and iterations than bidiagonalization and pre preconditioned bidiagonalization technique. But the nonlinearized S.I.P. technique required less time and iterations than post preconditioned bidiagonalization technique.

5. In the early 70's a new iterative technique was proposed by A. Brandt called multilevel adaptive technique or multigrid method, and developed by several authors (e.g. Hackbush (H1)) using different iterative techniques as a relaxation method (e.g. incomplete LU-decomposition (H10), Newton's method (F6)).

In Chapter 6 we solved a laminar flow problem (as an example of unsymmetric linear or non linear system) using a nonlinearized S.I.P. technique as a relaxation method with multigrid acceleration. Numerical results show, compared with the results in Chapter 5, that solving linear or non linear system using multigrid method with nonlinearized S.I.P. as a relaxation technique, requires less time and iterations than the iterative technique nonlinearized S.I.P.

Several suggestions for the development of the work in this thesis can be performed, for instance:

1) In the field of the approximate matrix factorization, several factorizations have been proposed, so it is worthwhile to make a complete survey and comparative study on the performance and the efficiency of different approximate matrix factorization to solve a system of non linear equations using nonlinearized S.I.P. and preconditioned non linear conjugate gradient method.

2) Bidiagonalization technique can be generalized to solve a system of non linear equations, and the suggested algorithm is as follows:

Non Linearized Bidiagonalization Algorithm

1. $\tau_0 = 1$, $\omega_0 = 0$, $\eta_0 = -1$, $vz = 0$, $v\omega = 0$

   $\beta_1 u_1 = R$ (R is the current residual)

   $\alpha_1 v_1 = J_1^T u_1$

2. Iteration i

   $\eta_i = -\eta_{i-1}\beta_i/\alpha_i$ $\qquad vz = vz + \eta_i v_i$

   $\omega_i = (\tau_{i-1} - \beta_i\omega_{i-1})/\alpha_i$ $\qquad v\omega = v\omega + \omega_i v_i$

   $\beta_{i+1} u_{i+1} = J_i v_i - \alpha_i u_i$

   If $\beta_{i+1} = 0$, then (solution = vz, residual = 0, STOP)

   $\tau_i = \tau_{i-1}\alpha_i/\beta_{i+1}$

   $\alpha_i v_{i+1} = J_i^T u_{i+1}/\beta_{i+1} v_i$

   If $\alpha_{i+1} = 0$, then ($\gamma = \beta_{i+1}\eta_i/(\beta_{i+1}\omega_i - \tau_i)$, solution = $vz - \gamma v\omega$, STOP)

3. Otherwise, set i = i + 1, and go to 2

   (J_i is the Jacobian of the non linear system at the i-th step of iteration)

3) Also it is possible to generalize the "Post Preconditioned Biadiagonalization Technique" to solve a system of non linear equations and to prove the performance of the following technique.

Non Linearized Post Preconditioned Bidiagonalization Algorithm

1. $\tau_0 = 1$, $\omega_0 = 0$, $\eta_0 = 1$, $vz = 0$, $v\omega = 0$

   $\beta_1 u_1 = R$ (R is the current residual)

   solve $C_1^T u_1^* = J_1^T u_1$

   $\alpha_1 v_1 = u_1^*$

2. Iteration i

   $\eta_i = -\eta_{i-1}\beta_i/\alpha_i$ $\qquad vz = vz + \eta_i v_i$

   $\omega_i = (\tau_{i-1} - \beta_i\omega_{i-1})/\alpha_i$ $\qquad v\omega = v\omega + \omega_i v_i$

   solve $C_i v_i^* = v_i$

   $\beta_{i+1} u_{i+1} = J_i v_i^* - \alpha_i u_i$

   If $\beta_{i+1} = 0$, then (solution = vz, residual = 0, STOP)

   $\tau_i = \tau_{i-1}\alpha_i/\beta_{i+1}$

   solve $C_i^T u_{i+1}^* = J_i^T u_{i+1}$

   $\alpha_{i+1} v_{i+1} = u_{i+1}^* - \beta_{i+1} v_i$

   If $\alpha_{i+1} = 0$, then ($\gamma = \beta_{i+1}\eta_i / (\beta_{i+1}\omega_i - \tau_i)$, solution = $vz - \gamma v\omega$, STOP)

3, Otherwise, set i = i + 1, and go to 2

   (J_i is the Jacobian of the system of the non linear equations, and C_i is the approximate Jacobian factorization, at the i-th step of iteration).

As it has been pointed out that the post preconditioned bidiagonalization technique is to solve the non symmetric linear (or non linear) system

of equations, it is worthwhile to make a survey on the performance of different approximate non symmetrix approximate matrix factorization using non linear post preconditioned bidiagonalization technique.

4. To study theoretically and analytically different iterative techniques (e.g. preconditioned non linear conjugate gradient method, post preconditioned bidiagonalization technique) as a relaxation technique in multigrid method).

REFERENCES

A1 A.N. Ahmed and M.C. Chen: On matrix factorization algorithm, Intern. J Computer Math., Vol.6, 1978, pp.305-317.

A2 A.C.Aitken: On the iterative solution of a system of linear equations, Proc. Royal Soc. Edinburgh, Vol.52, 1967, pp.52-60.

A3 P. Alfeld: A special class of explicit linear multistep method on basic methods for the correction in the dominant space technique, Math. Comp., Vol.33, No.148, 1979, pp.1195-1212.

A4 W.F. Ames: Non-linear partial differential equations in engineering, Academic Press, 1965.

A5 O. Axelsson: A generalized SSOR method, BIT, Vol.13, 1972, pp.440-467.

A6 O. Axelsson: On preconditioning and convergence acceleration in sparse matrix problems, CERN European Organization for Nuclear Research Rep. No. 74-10, 1974.

A7 O. Axelsson: On preconditioned conjugate gradient methods, Harwell Report. Rep. No. AERE-R9636, 1979.

A8 O. Axelsson: A class of iterative methods for finite element equations, Computer Methods in Applied Mechanics and Engineering, Vol.9-1976, pp.123-137.

B1 R. Balu: An application of Keller's method to the solution of an eight order non linear boundary value problem, Inter. J. Numer. Methods in Engineering, Vol.15, 1980, pp.1177-1180.

B2 R. Bartles and J.W. Daniel: A conjugate gradient approach to non-linear elliptic boundary value problem in irregular regions, Proced. of the Conference on the Numerical Solution of Differential Equations, Dundee Conference, pp.1-11, Springer Verlag, 1974.

B3 V. Berni and D. Fort-Urato: Some non-linear elliptic problems with asymptotic conditions, Non-linear Analysis Theory, Methods and Applications, Vol.3, No.2, 1979, pp.157-174.

B4 L. Bers: On mildly non-linear partial differential equations of elliptic type, J. Res. The National Bureau of Standards, Vol.51, No.5, 1953, pp.229-237.

B5 G. Birkhoff and R.S. Varga: Implicit alternating direction method, Trans. Amer. Math. Soc., Vol.92, 1959, pp.13-24.

B6 A. Bjork: Methods for linear least squares problems,
The Numerical Solution of Elliptic Partial Differential Equations,
ed. J.R. Bunch and D.J. Rose, Academic Press, 1974.

B7 A. Bjork: Use of conjugate gradients for solving linear least squares problems,
AERE, Harwell, R-9636, 1979.

B8 P.T. Boggs: The solution of non-linear systems of equations by A-stable integration techniques,
SIAM J. Numer. Anal., Vol.8, No.4, 1971, pp.767-784.

B9 P.T. Boggs and J.E. Dennis: A stability analysis for perturbed non-linear iterative methods,
Math. Comp., Vol.30, No.134, 1976, pp.199-215.

B10 A. Bracha-Barak and P.E. Saylor: Asymptotic factorization procedure for the solution of elliptic boundary value problems,
SIAM J. Numer. Anal., Vol.10, 1973, pp.198-206.

B11 J.M. Bramble and B.E. Hubbard: A theorem on error estimation finite difference analysis analogues of the Dirichlet problem for elliptic equations,
Contributions to Differential Equations, Vol.2, 1963,
Academic Press.

B12 J.M. Bramble and B.E. Hubbard: On the formulation of finite difference of the Dirichlet problem for Poisson's equations,
Numer. Math., Vol.4, 1962, pp.313-327.

B13 A. Brandt: Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems,
Lecture Notes in Physics, 1973, pp.82-89.

B14 A. Brandt: Multi level adaptive solutions to boundary value problems,
Math. Comp., Vol.31, No.138, 1977, pp.333-390.

B15 A. Brandt: Multi level adaptive techniques (MLAT) for partial differential equations, ideas and software,
Mathematical Software, Vol.III, Ed. J. Rice, 1977.

B16 A. Brandt: Multi level adaptive technique for singular perturbation problem,
Numerical Analysis of Singular Perturbation Problems,
Ed. P.W. Hemker and J.J.H. Miller, Academic Press, 1979.

B17 A. Brandt: Multi level adaptive computations in fluid dynamics,
Weizman Inst. of Science. (Special Report)

B18 A. Brandt: Numerical stability and fast solution to boundary value problems,
Weizman Inst. of Science Rep., 1979.

B19 A. Brandt and N. Dinar: Multigrid solution to elliptic flow problems,
Numerical methods for PDE's, 1979. (Special Report)

B20 J. Brauninger: A quasi-Newton method for minimization under linear constraints without evaluation any derivatives, Computing, Vol.12, 1979, pp.127-141.

B21 R.K. Brayton, F.G. Custavsenon and R.A. Willoughby: Results on sparse matrices, Math. Comput., Vol.24, 1970, pp.937-954.

B22 R.P. Brent: Some efficient algorithms for solving systems of non-linear equations, SIAM J. Numer. Anal., Vol.10, 1973.

B23 K.M. Brown and W.B. Gearhart: Deflation techniques for the calculation of further solutions of a non-linear system, Numer. Math, 1971, pp.334-342.

B24 N.J. Buleev: A numerical method for the solution of two dimensional and three dimensional equations of diffusion, Math.Sb., Vol.51, 1960, pp.227-238.

B25 J.R. Bunch: Partial pivoting: strategies for symmetric matrices, SIAM J. Numer. Anal., Vol.11, 1974, pp.521-528.

C1 H. Canh: A convergence theorem on the iterative solution of non-linear two point boundary value systems, Kybernetika, Vol.10, 1974, pp.49-60.

C2 R. Chandra: Conjugate gradient for partial differential equations, Yale University, 1978, Thesis.

C3 J. Chavette and F. Stenger: The approximate solution of the non-linear equation $\Delta U = U - U^3$, J. of Mathematical Analysis and Applications, Vol.15, 1975, pp.229-242.

C4 P.G. C iarlet, M.H. Schultz and R. Varga: Numerical methods of high order accuracy for non linear boundary value problems I, Numer. Math., Vol.9, 1967, pp.394-430.

C5 A.K. Cline et al: An estimate for the condition number of A-matrix SIAM J. Numer. Anal., Vol.16, No.2, 1979, pp.368-375.

C6 P. Concus: Numerical solution of the minimal surface equation, Math. Comp., Vol.21, 1967, pp.340-350.

C7 P. Concus: Numerical solution of the minimal surface equation by block non-linear SOR, Information Processing, Vol.68, 1969, pp.153-158.

C8 P. Concus and G.H. Golub: A generalized conjugate gradient method in applied science and engineering, IRIA, 1975-1976, pp.56-65.

C9 P. Concus, G.H. Golub and D.P. O'Leary: A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations, Sparse Matrix Computations, Ed. J.R. Bunch and D.J. Rose, 1976, pp.309-333.

C10 P. Concus, G.H. Golub and D.P. O'Leary: Numerical solution of non linear elliptic partial differential equations by a generalized conjugate gradient method,
Computing, Vol.19, 1978, pp.321-339.

C11 A.R. Curtis and J.K. Reid: The solution of large sparse unsymmetric systems of linear equations,
J. Inst. Maths. Applics., Vol.8, 1971, pp.344-353.

D1 G. Dahlquist and A. Bjork: Numerical methods.
Prentice Hall Inc., 1975.

D2 J.W. Daniel: The conjugate gradient method for linear and non-linear operator equations,
SIAM J. Numer. Anal., Vol.4, No.1, 1967, pp.10-26.

D3 J.W. Daniel: Convergence of the conjugate gradient method with computationally convenient modifications,
Numer. Math., Vol.10, pp.125-131, 1967.

D4 F. De la Valle Poussin: An accelerated relaxation algorithm for iterative solution of elliptic equations,
SIAM J. Numer. Anal., Vol.5, 1968, pp.340-351.

D5 J.E. Dennis, Jr.: Non linear least squares and equations,
"The State of the Art in Numerical Analysis", Ed. D. Jacobs, 1977.

D6 J.E. Dennis, Jr.: Some computational techniques for the non linear equations,
Ed. G.B. Byrne and C.A. Hall, Academic Press, 1973.

D7 J.E. Dennis and J.J. More: Quasi-Newton methods motivation and theory,
SIAM Review, Vol.19, 1977, pp.46-88.

D8 S.K. Dey: Numerical solution of a system of non-linear equations by perturbation,
Demonstratio Mathematics, Vol.9, No.4, 1976, pp.69-705.

D9 J. Douglas, Jr.: A note on the alternating direction implicit method for the numerical solution of heat conduction problem,
Proc. Amer. Math. Soc., Vol.8, pp.409-412, 1957.

D10 J. Douglas, Jr.: Alternating direction iteration for mildly non-linear elliptic difference equations,
Numer. Math., Vol.3, pp.92-98, 1961.
Numer. Math., Vol.4, pp.301-302, 1962.

D11 J. Douglas, Jr., T. Dupont and J. Serrin: Uniqueness and comparison theorems for non-linear elliptic equations in divergence form,
Arch. National Mech. Anal., Vol.42, pp.157-168, 1971.

D12 A.A. Dosiev and J.D. Mamedov: On the solution by the grid method of a mixed boundary value problem for non-linear elliptic equations,
Dokl. Akad. Nauk SSSR, Tom 242, No.4, 1978.
Soviet Math. Dokl., Vol.19, No.5, 1978, pp.1186-1189.

D13 T. Dupont: A factorization procedure for the solution of elliptic difference equations,
SIAM J. Numer. Anal., Vol.5, No.4, 1968, pp.753-782.

D14 T. Dupont et al: An approximate factorization for solving self adjoint elliptic difference equations,
SIAM J. Numer. Anal., Vol.5, No.3, 1968, pp.554-573.

E1 L.W. Ehrlich: The block symmetric successive over relaxation method,
J. Soc. Indust. Appl. Math., Vol.12, pp.807-826, 1964.

E2 S.C. Eisenstat, M.H. Schultz and A.H. Sherman: The application of sparse matrix methods to the numerical solution of non-linear partial differential equations,
Lecture notes in Mathematics, Vol.430, Springer Verlag.

E3 D.J. Evans: The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices,
J. Inst. Maths. Applics., Vol.4, 1967, pp.295-314.

E4 D.J. Evans: On the numerical solution of sparse system of finite element equations,
MAFELAB, 1978, Ed. J.R. Whiteman.

F1 R.P. Redorenko: A relaxation method for solving elliptic difference equations,
USSR Comp. Maths and Mathematical Physics, pp.1092-1096, 1966.

F2 R. Fletcher and C.M. Reeves: Function minimization by conjugate gradient,
Dundee Conference on the Numerical Solution of Differential Equation 1973, Lecture Notes in Mathematics, 1974.

F3 L. Fox: Finite difference methods for elliptic boundary value problems,
The State of Art in Numerical Analysis, Ed. D. Jacobs, 1977.

F4 P.O. Frederickson: Fast approximate inversion of large sparse linear systems,
Dept. of Mathematical Sciences, Lakehead University Rep. 7-75, 1975.

F5 E. Freudenstein and B. Roth: Solution of a system of non-linear equations,
J. Association Comput. Machinery, Vol.10, pp.550-556, 1963.

F6 L. Fuchs: A Newton's multigrid method for the solution of non-linear partial differential equations,
Proceeding of the BAIL 1 Conference, Ed. J.J.H. Miller, 1980.

G1 G. Gamolti: Fast solution to finite element flow equation by Newton iteration and modified conjugate gradient method, Intr . J. Numer. Methods in Engineering, Vol.15, 1980, pp.661-675.

G2 M. Ganh: A convergence theorem on the iterative solution of non-linear twopoint boundary value systems, Kyberretrika, Vol.10, 1974.

G3 G. Gheri and O.G. Mancino: A significant example to test methods for solving systems of non-linear equations, CALCOLO, Vol.8, 1971, pp.107-113.

G4 P.E. Gill et al: Methods for modifying matrix factorization, Math. Comp., Vol.25, 1974, pp.505-535.

G5 P.E. Gill and W. Murray: Algorithms for the solution of the non-linear least squares, SIAM J. Numer. Anal., Vol.15, No.5, 1978, pp.977-992.

G6 G. Golub and W. Khan: Calculating the singular values and pseudo-inverse of a matrix, SIAM J. Numer. Anal., Vol.2, 1965, pp.205-224.

G7 D. Greenspan and S.V. Parter: Mildly non linear elliptic partial differential equations and their numerical solution II, Numer. Math., Vol.7, 1965, pp.129-140.

G8 J.E. Gunn: The numerical solution of $\nabla . a \nabla u = f$, by semi explicit alternating direction iterative technique, Numer. Math., Vol.6, 1964, pp.181-184.

H1 W. Hackbush: A fast numerical method for solving poissons equation in a general region, Numerical Treatment of Differential Equations, 1976, Lecture Notes in Mathematics, 1978.

H2 W. Hackbush: A multi grid method applied to a boundary value problem with variable coefficients in a rectangle, Rep.No. 77-17, 1977.

H3 W. Hackbush: On the multi grid method applied to difference equations, Comput., Vol.20, 1978, pp.291-300.

H4 W. Hackbush: On the fast solving of elliptic control problems, Rep. 78-14, 1978.

H5 W. Hackbush: An error analysis of the non linear multi grid method of second kind, Report 78-15, 1978.

H6 W. Hackbush: On the fast solutions of non linear elliptic equations, Numer. Math., Vol.32, 1979, pp.83-95.

H7 W. Hackbush: Convergence of multi grid iterations applied to difference equations, Math. Comp., Vol.34, 1980, pp.425-440.

H8 L.A. Hageman and T.A. Porsching: Aspects of non linear block successive over relaxation,
SIAM J. Numer. Anal., Vol.12, No.3, 1975, pp.316-335.

H9 L. Hayes and E. Wasserstron: Solution of non linear eigenvalue problems by continuation method,
J. Inst. Maths. Applics., Vol.17, 1976, pp.5-14.

H10 P.W. Hemker: The incomplete LU-decomposition as a relaxation method in multigrid algorithms,
Proced. of the BAIL 1 Conference, Ed. J.J.H. Miller, 1980.

H11 R.J. Herbold and M. Schultz and R. Varga: The effects of quadrature errors in the numerical solution of boundary value problems by variational techniques,
AECUATIONES MATH. S., Vol. 3, 1969, pp.247-270.

H12 M.R. Hestenes and E. Stiefel: Methods of conjugate gradients for solving linear systems,
J. of Research of National Bureau of Standards, Vol.47, 1952, pp.409-438.

H13 J.M. Hyman: Mesh refinements and local inversion of elliptic partial differential equations,
J. of Comp. Physics, Vol.23-1977, pp.124-134.

J1 D.A.H. Jacobs: The strongly implicit procedure for biharmonic problems,
J. of Computational Physics, Vol.13, 1973, pp.303-315.

J2 P.C. Jain and M.K. Kadalbajou: Dynamic programming solutions of mildly non-linear elliptic problems over irregular regions,
Intern. J. Computer Math., Sect.B, Vol.5, 1976, pp.231-239.

J3 P. Jarratt and N. Parkin: On the asymptotic behaviour of a certain non-linear difference equation,
J. Inst. Maths. Applics., Vol.13, 1974, pp.83-88.

J4 A. Jennings: Acceleration the convergence of matrix iterative processes,
J. Inst. Maths. Applics., Vol.8, 1971, pp.99-110.

J5 A. Jennings and G.M. Malik: The solution of sparse linear equations by the conjugate gradient method,
Intr. J. for Numerical Methods in Engineering, Vol.12, 1978, pp.141-158.

J6 A. Jennings and G.M. Malik: Partial elimination,
J. Inst. Maths. Appl., Vol.20, 1977, pp.307-316.

K1 M.M. Karcherski and A.D. Lyashke: Difference schemes for quasi-linear elliptic equations with discontinuous coefficients,
Soviet Math., Vol.18, No.5, 1974, pp.128-137.

K2 H.B. Keller: Approximation methods for non-linear problems with applications to two points boundary value problems,
Math. Comp., Vol.29, 1975, pp.464-474.

K3 H.B. Keller: Elliptic boundary value problems suggested by non-linear diffusion,
ARCHIVE for Rational Mechanics and Analysis, Vol.35, 1969.

K4 D.S. Kershaw: The incomplete cholesky conjugate gradient method for the iterative solution of systems of linear equations,
J. of Computational Physics, Vol.26, 1978, pp.43-65.

L1 C.Lanczos: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators,
J. of Research of the National Bureau of Standards, Vol.45, 1950, pp.1247-1252.

L2 L.S. Lasdon: Conjugate direction methods for optimal control,
IEEF. Trans. Automatic Control, Vol.15, 1970, pp.267-268.

L3 A.S. Leonor: On the construction of stable difference schemes for solving non-linear boundary problems,
Dokl. Akad. Nauk SSSR, Tom 224, 1975.

L4 N. Levinson: Dirichlet problem for $\Delta u = f(p,u)$,
J. of Math. and Mech., Vol.12, 1963, pp.567-576.

L5 S.A. Lill: A survey of methods for minimizing sums of squares of non linear fractions,
Proc. of Conference Optimization in Action, Bristol University, Ed. Dixon, 1976-77, Academic Press.

L6 E.A. Lipitakis and D.J. Evans: Solving non linear elliptic difference equations by extendable sparse factorization procedures,
Computing, Vol.24, 1980, pp.325-339.

L7 L.R. Lundin: A cardinal function method of solution of the equation $\Delta u = u - u^3$,
Maths. Comp., Vol.35,1980, pp.747-756.

M1 K. Madson: On the solution of non-linear equations in internal arithmetic,
BIT, Vol.13, 1973, pp.428-433.

M2 T.A. Manteuffel: An incomplete factorization technique for positive definite linear systems,
Math. Comp., Vol.34, No.150, 1980, pp.473-497.

M3 G.T. McAllister: Some non linear elliptic partial differential equations and difference equations,
J. Soc. Indust. App. Math., Vol.12, 1964, pp.772-777.

M4 G.T. McAllister: Newton's method for non-linear elliptic problems,
Math. Nachr., 1971, pp.25-33.

M5 G.T. McAllister: Quasi-linear uniformly elliptic partial differential equations and diffreence equations,
SIAM J. Numer. Anal., Vol.3, 1966, pp.13-33.

M6 C. McCarthy and G. Strong: Optimal conditioning of matrices, SIAM J. Numer. Anal., Vol.10, 1973, pp.370-388.

M7 J.A. Meijerink and H.A. Van der Vorst: An iterative solution method for linear systems of which the coefficient matrix is symmetric M-matrix, Math. Comp., Vol.31, 1977, pp.148-162.

M8 J.A. Meijerink and H.A. Van der Vorst: Incomplete decompositions as a preconditioning for the conjugate gradient algorithm, Harwell, AERE Report No. 9636, 1979.

M9 M. Meninger: Sparse matrix reduction, Telsa Electronics, Vol.6, 1973, pp.48-73.

M10 G.H. Meyer: The numerical solution of quasi-linear elliptic equation, Numerical solution of systems of non-linear algebraic equations, Ed. G.D. Byrne and C.A. Hall, Academic Press, 1973.

M11 R. Meyer-Spasche: A note on the approximation of mildly non-linear Dirichlet problems by finite difference, Numer. Math., Vol.33, 1979, pp.303-313.

M12 H.D. Mittelmann: On pointwise estimates for a finite element solution of non linear boundary value problems, SIAM J. Numer. Anal., Vol.14, 1977, pp.773-778.

M13 R.E. Moore: A test for existence of solutions to non linear system, SIAM J. Numer. Anal., Vol.14, 1977, pp.611-615.

M14 R.E. Moore: A computational test for convergence of iterative methods for non linear systems, SIAM J. Numer. Anal., Vol.15, 1978, pp.1194-1196.

M15 J.J. More: Global convergence of Newton-Gauss Seidel methods, SIAM J. Numer. Anal., Vol.8, 1971, pp.325-330.

M16 H. Munksgard: Solving sparse symmetric linear equations by pre-conditioned conjugate gradients, Harwell Rep. CS-67, 1979.

N1 K. Nakashima: Difference analogue for non linear elliptic partial differential equations with high order of accuracy, MEMORIRS of the School of Science and Engineering, Waseda University, No.32, 1968, pp.77-86.

N2 L. Nazareth: Some recent approaches for solving large residual non linear least squares problems, SIAM Review, Vol.22, 1980, pp.1-11.

N3 C.P. Neuman and A. Sen: Galerkin's procedure, quasi-linearization and non-linear boundary value problems, J. of Optimization, Theory and Applications, Vol.9, 1972, pp.433-437.

N4 R.A. Nicolaides: On the local convergence of certain two step iterative procedures,
Numer. Math., Vol.24, 1975, pp.95-101.

N5 R.A. Nicolaides: On multiple grid and related techniques for solving discrete elliptic systems,
J. of Computational Physics, Vol.19, 1975, pp.418-431.

N6 R.A. Nicolaides: On finite element multigrid algorithms and their use,
MAFELAB, 1978, Ed. J.R. Whiteman.

N7 R.A. Nicolaides: On the ℓ^2-convergence of an algorithm for solving finite difference equations,
Math. Comp., Vol.31, 1977, pp.892-906.

N8 R.A. Nicolaides: On multigrid convergence in the indefinite case,
Math. Comp., Vol.32, No.144, 1978, pp.1082-1086.

N9 R.A. Nicolaides: On the observed rate of convergence of an iterative method applied to a model elliptic difference equation,
Math. Comp., Vol.32, 1978, pp.127-133.

N10 R.A. Nicolaides: On some theoretical and practical aspects of multigrid methods,
Math. Comp., Vol.33, 1979, pp.933-952.

N11 W. Niethammer and J. Schade: On relaxed SOR-method applied to non-symmetric linear systems,
J. of Comp. and Applied Mathematics, Vol.1, 1975, pp.133-136.

N12 M.A. Noor and J.R. Whiteman: Error bounds for finite element solutions of mildly non-linear elliptic problems,
Numer. Math., Vol.26, 1976, pp.107-116.

O1 T.A. Oliphant: An extrapolation procedure for solving linear systems,
Quart. Appl. Math., Vol.20, 1962, pp.257-267.

O2 J.M. Ortega: Stability of difference equations and convergence of iterative processes,
SIAM J. Numer. Anal., Vol.10, 1973, pp.268-282.

O3 J.M. Ortega and M.L. Rockoff: Non linear difference equations and Gauss-Seidel type iterative methods,
SIAM J. Numer. Anal., Vol.3, 1966, pp.497-513.

O4 J.M. Ortega and W.C. Rheinboldt: Monotone iterations for non linear equations with applications to Gauss-Seidel methods,
SIAM J. Numer. Anal., Vol.4, 1967, pp.171-189.

O5 J.M. Ortega and W.C. Rheinboldt: Iterative solution of non linear equations in several variables,
Academic Press, 1970.

O6 J.M. Ortega and W.C. Rheinboldt: A general convergence result for unconstrained minimization methods,
SIAM J. Numer. Anal., Vol.9, 1972, pp.40-43.

P1 C.C. Paige: Bidiagonalization of matrices and solution of linear equations,
SIAM J. Numer. Anal., Vol.11, 1974, pp.197-208.

P2 C.C. Paige: Fast numerical stable computations for generalized linear least squares problems,
SIAM J. Numer. Anal., Vol.16, 1979, pp.165-171.

P3 C.C. Paige: Computational and variants of the Lanczo's method for the eigen problem,
J. Inst. Maths. Applics., Vol.10, 1972, pp.373-381.

P4 C.C. Paige and M.A. Saunders: Solutions of sparse indefinite systems of equations,
SIAM J. Numer. Anal., Vol.12, No.4, 1975, pp.617-629.

P5 C.C. Paige and M.A. Saunders: Least squares estimations of discrete linear dynamics system using orthogonal transformations,
SIAM J. Numer. Anal., Vol. 14, No.2, 1977, pp.180-193.

P6 S.V. Parter: Mildly non-linear elliptic partial differential equations and their numerical solution I,
Numer. Math., Vol.7, 1965, pp.113-128.

P7 V. Pereyra: Iterative methods for solving non-linear least squares problem,
SIAM J. Numer. Anal., Vol.4, No.1, 1967, pp.27-30.

P8 S.I. Pohozaer: The Dirichlet problem for the equation $\Delta U = U^2$,
Soviet Mathematics, Vol.12, pp.1143-1146.

P9 T.A. Porsching: Jacobi and Gauss Seidel methods for non-linear network problems,
SIAM J. Numer. Anal., Vol.6, No.3, 1969, pp.437-449.

P10 T.A. Porsching: On the origins and numerical solutions of some sparse non-linear systems,
Sparse Matrix Computations, Ed. J.R. Bunch and D.J. Rose,
Academic Press, 1976.

R1 J.K. Reid: On the method of conjugate gradients for the solution of large sparse system of linear equations,
Proc. Conf. on "Large Sparse Sets of Linear Equations",
Academic Press, 1971.

R2 J.K. Reid: The use of conjugate gradients for systems of linear equations processing "Property A",
SIAM J. Numer. Anal., Vol.9, No.2, 1972, pp.325-332.

R3 W.C. Reinboldt: On methods for solving systems of non-linear equations,
Regional Conference Series in Applied Mathematics, No.14,
Published by SIAM, Philadelphia.

R4 W.C. Reinboldt: On the measure of ill-conditioning for non-linear equations,
Math. Comp., Vol.30, 1976, pp.104-111.

R5 W.C. Reinboldt: Solution fields of non-linear equations and continuation methods,
SIAM J. Numer. Anal., Vol.17, pp221-287, 1980.

R6 W.C. Reinboldt and J.S. Van der Graft: On the local convergence of update methods,
SIAM J. Numer. Anal., Vol.11, 1974, pp.1069-1085.

S1 P.S. Saylor: Second order strongly implicit symmetric factorization methods for the solution of elliptic difference equations,
SIAM J. Numer. Anal., Vol.11, 1974, pp.894-908.

S2 H. Schomberg: Monotically convergent iterative methods for non-linear systems of equations,
Numer. Math., Vol.32, 1979, pp.97-104.

S3 N.L. Schryer: Newton's method for convex non-linear elliptic boundary value problems,
Numer. Math., Vol.17, 1971, pp.284-300.

S4 N.L. Schryer: Solution of monotone non-linear elliptic boundary value problem,
Numer. Math., Vol.18, 1972, pp.336-344.

S5 G. Schuller: On the order of convergence of certain quasi-Newton methods,
Numer. Math., Vol.23, 1974, pp.181-192.

S6 L.F. Shampine: Error bounds and variational methods for non linear boundary value problems,
Numer. Math., Vol.12, 1968, pp.410-415.

S7 D.F. Shanno: On the convergence of new conjugate gradient algorithm,
SIAM J. Numer. Anal., Vol.15, 1978, pp.1247-1257.

S8 A.H. Sherman: On Newton-iterative methods for the solution of systems of non-linear equations,
SIAM J. Numer. Anal., Vol.15, No.4, 1978, pp.755-771.

S9 S.M. Shugrin: An iteration method,
SIBERIAN Math. J., Vol.12, 1971, pp.490-498.

S10 R.B. Simpson: Finite difference methods for mildly non-linear eigenvalue problems,
SIAM J. Numer. Anal., Vol.8, 1971, pp.190-211.

S11 R.B. Simpson: Existence and error estimates for solutions of a discrete analog of non-linear eigenvalue problems,
Math. Comp., Vol.26, No.118, 1972,pp.359-375.

S12 J.C. South, Jr. and A. Brandt: Application of a multi-level grid method to transonic flow calculators, ICASE Rep. 1976.

S13 G. Starius: Composite mesh difference methods for elliptic boundary value problems, Numer. Math., Vol.28, pp.243-258, 1977.

S14 R.S. Stepleman: Difference analogues of quasi-linear elliptic Dirichlet problems with mixed derivatives, Math. Comp., Vol.25, No.114, 1971, pp.257-269.

S15 H.L. Stone: Iterative solution of implicit approximations of multi-dimensional partial differential equations, SIAM J. Numer. Anal., Vol.5, 1968, pp.530-558.

T1 R.P. Tewarson: A unified derivation of quasi-Newton methods for solving non sparse and sparse non linear equations, Computing, Vol.21, 1979, pp.113-125.

T2 K. Thews: A reduction method for some non linear Dirichlet problems, Nonlinear Analyais, Theory, Methods and Applications, Vol.6, 1979, pp.795-813.

T3 A.D. Tuff and A. Jennings: An iterative method for large systems of linear structural equations, Inter. J. Numerical Methods in Engineering, Vol.7, 1973, pp.175-183.

V1 R.S. Varga: Matrix iterative analysis, Prentice Hall Inc., 1962.

V2 R.G. Voigt: Rates of convergence for a class of iterative procedures, SIAM J. Numer. Anal., Vol.8, 1971, pp.127-130.

V3 R.G. Voigt: Order of convergence for iterative procedures, SIAM J. Numer. Anal., Vol.8, 1971, pp.222-243.

W1 E.L. Wachspress: Iterative solution of elliptic systems, Prentice Hall Inc., 1966.

W2 S.E. Weinstein: Solution of non linear equations by iterative procedures which use approximation techniques, SIAM J. Numer. Anal., Vol.6, 1969, pp.272-283.

W3 P. Wesseling: The rate of convergence of a multiple grid method, Rep. NA-30, Delft, 1979.

W4 J.H. Wilkinson: The algebraic eigenvalue problem, Oxford University Press, 1965.

W5 G.A. Wilson and T.R. Rogge: Solutions of a system of non linear equations, J. Indust. Mathematics Society, Vol.19, 1969, pp.1-14.

W6 M.A. Wolfe: Some methods for least squares estimation,
J. Inst. Maths. Applics., Vol.18, 1976, pp.219-230.

W7 Y.S. Wong: Preconditioned conjugate gradient methods for biharmonic problems,
Dundee Conference, 1979.

W8 H. Wozniakowski: Numerical stability for solving non linear equations,
Numer. Math., Vol.27, 1977, pp.373-390.

Y1 D. Young: Iterative solution of large linear systems,
Academic Press, 1971.

Y2 D.M. Young and W.F. Wheeler: Alternatively direction methods for solving partial difference equations,
Non Linear Problems of Engineering, Ed. W.F. Ames, Academic Press, 1964.